12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics

SCAN 2006

September 26-29, 2006, Duisburg, Germany

Conference Post-Proceedings

Foreword

12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics

Duisburg, Germany, 26-29 September 2006

This conference continued the series of international SCAN symposia initiated by the University of Karlsruhe. SCAN symposia have been held in many towns in Europe and Japan under the joint sponsorship of GAMM and IMACS: Basel (1989), Varna-Albena (1990), Oldenburg (1991), Vienna (1993), Wuppertal (1995), Lyon (1997), Budapest (1998), Karlsruhe (2000), Paris (2002), Fukuoka (2004) and Duisburg (2006).

Scope of the Conference

SCAN symposia have traditionally covered the numerical and algorithmic aspects of scientific computing, with a strong emphasis on how validation and verification provide guaranteed properties of computed results as well as on arithmetic, programming and algorithmic tools for this purpose.

They present a state-of-the-art overview on the challenging and dynamic field of reliable computing techniques and interval arithmetic for the researchers, experts and scientists who apply these techniques. Contributions provide competent and concise information on recent hardware and software standards, language support with interval arithmetic, algorithms with result verification and applications in various fields.

An important part of the 2006 conference was devoted to software systems for verified numerics using enhanced interval arithmetic, appropriate data types and accurate numerical, geometric or stochastic modeling. Other tools for validation and avoiding round-off errors are based on exact or formal representation or include symbolic, algebraic and algorithmic methods. One of the missions of SCAN 2006 was to provide a forum for presenting the many validation approaches currently in use and for serious debate about scientific computing and result verification.

Scientific Program

The program consists of invited plenary lectures and contributed presentations. Invited speakers and presentations:

Tibor Csendes, University of Szeged: *Global Optimization and Verified Numerical Techniques for the Solution of Mathematical Problems*

Peter Hertling, Hochschule der Bundeswehr München: *Solvability and Bit Complexity of Problems Concerning Real Numbers*

Eberhard P. Hofer, Universität Ulm: *Applications of Interval Algorithms in Engineering*

Philippe Langlois, Université de Perpignan: *Accurate Algorithms in Floating Point Arithmetic*

Kaori Nagatou, Kyushu University: *Validated Computation for Infinite-Dimensional Eigenvalue Problems*

Ned Nedialkov, McMaster University Canada: *Tools for ODEs and DAEs*

Arnold Neumaier, Universität Wien: *Computer-Assisted Proofs*

Siegfried Rump, TU Hamburg-Harburg: *Self-Validating Methods*

Contributed presentations in nine parallel sessions were invited to all twelve conference subjects.

- Hardware support for numerical validation
- Software support for numerical validation (standardization, controlled rounding, accurate operations and functions, programming languages, user-oriented interfaces, standardized data transfer, automatic translation into programs with result verification)
- Arithmetic (floating point with error bounds, interval, fuzzy interval, affine, Taylor models, exact real, stochastic)
- Supercomputing (parallel and grid computing)
- Verification numerics (ODE, PDE, DAE, [non]linear systems)
- Exact methods, computability, domain theory
- Computer-aided proofs (linear algebra, geometry, dynamical systems)
- Validation in logic, symbolic, algebraic and algorithmic calculus
- Numerical tools using enhanced interval arithmetic
- Validation in optimization problems and dynamical systems
- Validated modeling and simulation (Parameter estimation, control, task planning, networks)
- Industrial and scientific applications (financial simulation and optimization, mechanical design, robotics, aeronautics, measurement of physical constants, environmental engineering, chemical process simulation and control, computer graphics, computer-aided geometric design, astrophysics, biology, medicine)

Young researchers and Ph.D. candidates were especially encouraged to contribute to the conference.

This book contains the proceedings of the 2006 SCAN meeting in Duisburg, Germany. We present written versions of five invited plenary lectures and 36 carefully selected papers from the paper sessions, which included a total of 99 contributions. We would like to thank all the authors for providing us with their interesting contributions and the referees listed below for helping to enhance the papers.

Wolfram Luther
Werner Otten
Duisburg, Germany
April 2007

Invited Lectures

Applications of interval algorithms in engineering
Eberhard P. Hofer and Andreas Rauh

Operator dependant compensated algorithms
Philippe Langlois and Nicolas Louvet

Validated computation for infinite dimensional eigenvalue problems
Kaori Nagatou

Interval tools for ODEs and DAEs
Nedialko S. Nedialkov

Computer-assisted proofs
Arnold Neumaier

Regular Lectures

On the solution to numerical problems using stochastic arithmetic
René Alt, Jean-Luc Lamotte, and Svetoslav Markov

Toward validating a simplified muscle activation model in SmartMOBILE
Ekaterina Auer, Martin Tändl, Daniel Strohbach, and Andres Kecskeméthy

GRKLib: a guaranteed Runge Kutta library
Olivier Bouissou and Matthieu Martel

Hardware implementation of continued logarithm arithmetic
Tomás Brabec

Rigorous lower bounds for the topological entropy
via a verified optimization technique
Tibor Csendes, Balázs Bánhelyi, and Barnabás M. Garay

Nonlinear adaptive control of an uncertain wastewater treatment model
Neli Dimitrova and Mikhail Krastanov

Interval fuzzy rule-based hand gesture recognition
Gracaliz P. Dimuro, Benjamín R.C. Bedregal, and Antônio C. Rocha Costa

Analyzing properties of fuzzy implications obtained via the interval constructor
Gracaliz P. Dimuro, Benjamín R.C. Bedregal, Regivan H.N. Santiago, and Renata H.S. Reiser

Interval analysis of linear analog circuits
Alexander Dreyer

A reliable convex hull algorithm for interval-based hierarchical structures
Eva Dyllong

Parametric linear system of equations, whose elements are nonlinear functions
Hassan El-Owny

Guaranteed bounds for uncertain systems: methods using linear Lyapunov-like functions, differential inequalities and a midpoint method
Marc Gennat and Bernd Tibken

On the approximation of linear AE-solution sets
Alexandre Goldsztejn and Gilles Chabert

Rigorous inner approximation of the range of functions
Alexandre Goldsztejn and Wayne Hayes

Ensuring numerical quality in grid computing
Andreas Frommer and Matthias Hüsken

An interval version of the backward differentiation (BDF) method
Malgorzata Jankowska and Andrzej Marcinak

A workload analysis tool for discrete-time semi-Markovian servers
Sebastian Kempken

Efficient 16-bit floating-point interval processor for embedded systems and applications
Michel Kieffer, Stéphane Piskorski, Lionel Lacassagne, and Daniel Eti

Guaranteed robust tracking with flatness based controllers applying interval methods
Marco Kletting, Eberhard P. Hofer, and Felix Antritter

Interval observer design based on Taylor models for nonlinear uncertain continuous-time systems
Marco Kletting, Andreas Rauh, Eberhard P. Hofer, and Harald Aschemann

Strong unboundedness of interval linear programming problems
Jana Konícková

intpakX — an interval arithmetic package for Maple
Walter Krämer

Towards interval techniques for processing educational data
Vladik Kreinovich, Olga Kosheleva, Luc Longpré, Mourat Tschoshanov, and Gang Xiang

Towards combining probabilistic, interval, fuzzy uncertainty, and constraints: an example using the inverse problem in geophysics
Vladik Kreinovich, S.A. Starks, A.A. Velasco, M.G. Averill, R. Araiza, G. Xiang, and G.R. Keller

Adding constraints to situations when, in addition to intervals, we also have partial information about probabilities
Vladik Kreinovich, Martine Ceberio, Giang Xiang, Scott Ferson, and Cliff Joslyn

Fast and accurate multi-argument interval evaluation of polynomials
Andreas Frommer and Bruno Lang

Vectorised/semi-parallel interval multiplication
Eoin Malins, Marek Szularz, and Bryan Scotney

On the interval Gaussian algorithm
Günter Mayer

Numerical computation of the mapping degree using computational homology
Kansaku Nakakura and Sunao Murashige

Computer-assisted proofs in solving linear parametric problems
Evgenija D. Popova

ValEncIA-IVP: a comparison with other initial value problem solvers
Andreas Rauh, Eberhard P. Hofer, and Ekaterina Auer

Interval techniques for design of optimal and robust control strategies
Andreas Rauh, Johanna Minisini, and Eberhard P. Hofer

Deterministic global optimization for dynamic systems using interval analysis
Youdong Lin and Mark A. Stadtherr

Computing the Jordan canonical form in finite precision arithmetic
Toshio Suzuki and Tomohiro Suzuki

The fundamental theorems of interval analysis
M.H. van Emden and B. Moa

Expression defined accuracy
A. Pokorny and J. Wolff von Gudenberg

Applications of Interval Algorithms in Engineering

Eberhard P. Hofer and Andreas Rauh
Institute of Measurement, Control, and Microtechnology
University of Ulm
D-89069 Ulm, Germany
{EP.Hofer, Andreas.Rauh}@uni-ulm.de

Abstract

The optimization of the functionality and the guarantee of a safe operation of a technical system are important issues in industry. These aspects become even more important when we have to deal with numerous uncertainties which heavily influence the behavior of the technical system under consideration and — in the worst case — cause system failure. Appropriate interval tools can offer solutions to problems where system uncertainties play a key role. Over the recent years at the Institute of Measurement, Control, and Microtechnology existing interval tools have been extended and new modules have been developed.

In this contribution, successful applications of interval algorithms to real-world problems in various fields of engineering are presented. The focus is on measuring techniques including interval observers and sensitivity analysis as well as design of optimal and robust controllers for continuous-time and discrete-time systems.

1 Introduction

In this paper, an overview of various applications of interval algorithms in engineering is given[1]. A common basis of all applications is a given mathematical model of the relevant technical system described by sets of algebraic equations and ordinary differential equations. In general, two different types of problems can be considered which are steady state analysis and analysis of the transient behavior of a dynamical system.

For most practically relevant dynamical systems, guaranteed knowledge about the influence of uncertainties of both initial states and system parameters is of importance. In the following, different applications are presented in order to highlight the benefits of the use of interval methods [8, 13].

[1]For a complete list of recent applications of interval methods investigated by the authors, the reader is referred to `http://www.interval-methods.de`.

An application in automotive engineering covers the influence of unavoidable manufacturing errors on the functionality of a mechanical component. As a result, quality control in production can be improved. A further application deals with guaranteed estimation of physical parameters for characterization and model validation of a micro-electromechanical device.

Safety-critical applications in X-by-wire systems, e.g., automobiles and aircrafts, influenced by sensor uncertainties usually require the verification of the systems' functionality. Such uncertainties may not only affect the behavior of the feedback control, but also the monitoring of the function in the control unit and, thus, may lead to delayed or even false reactions in case of failure. Therefore, it is extremely important to get reliable results about the influence of sensor tolerances on the dynamic behavior of the closed-loop system. The task is formulated as a global interval optimization problem. It is solved by using advanced interval algorithms keeping all safety-critical states within a pre-defined limit.

In addition to safety aspects, reduction of operation costs of a plant is always a strong issue. In an application taken from environmental engineering the efficiency of interval methods not only for reliable plant operation but also for plant and controller design is shown.

Section 2 summarizes the notation used in this paper for the description of technical systems with uncertainties. The selected applications in engineering can be associated with the following problems:

- **Steady state analysis for time-invariant systems (Sec. 3).** The applications are a rocker arm (Sec. 3.1) and a micro relay (Sec. 3.2).
- **Analysis of *discrete*-time dynamical systems with time-*invariant* parameter uncertainties (Sec. 4).** The applications are an airbus elevator (Sec. 4.1) and a common-rail injection (Sec. 4.2).
- **Analysis of *continuous*-time dynamical systems with time-*varying* parameter uncertainties (Sec. 5).**

The application is a subsystem of biological wastewater treatment processes (Sec. 5.2).

- **State and parameter estimation using interval observers (Sec. 6).** The applications are an electrostatic microactuator (Sec. 6.1) and a micropositioning system (Sec. 6.2).

Section 7 summarizes the most important benefits that have been achieved by the use of interval methods for the applications presented here. An outlook on future research concludes this contribution.

2 Technical Systems with Uncertainties

The technical applications considered in this paper are described both by the discrete-time state-space representation

$$\begin{bmatrix} x_{k+1} \\ p_{k+1} \end{bmatrix} = \begin{bmatrix} g_k\left(x_k, p_k, u_k, k\right) \\ g_{p,k}\left(p_k, \Delta p_k\right) \end{bmatrix} \tag{1}$$

and — in the case of continuous-time processes — by ordinary differential equations

$$\begin{bmatrix} \dot{x}\left(t\right) \\ \dot{p}\left(t\right) \end{bmatrix} = \begin{bmatrix} f\left(x\left(t\right), p\left(t\right), u\left(t\right), t\right) \\ \Delta p\left(t\right) \end{bmatrix} \quad . \tag{2}$$

In both cases, uncertainties of the initial conditions of the state vector x have to be taken into account. For discrete-time models they are denoted by $x_0 \in [x_0] := [\underline{x}_0 \; ; \; \overline{x}_0]$, for continuous-time systems by $x\left(0\right) \in [x\left(0\right)] := [\underline{x}\left(0\right) \; ; \; \overline{x}\left(0\right)]$. In both system models, control vectors are denoted by u. All uncertain system parameters are represented by the parameter vector p which is bounded by the intervals $p_k \in \left[\underline{p}_k \; ; \; \overline{p}_k\right]$ for all $k \geq 0$ and $p\left(t\right) \in \left[\underline{p}\left(t\right) \; ; \; \overline{p}\left(t\right)\right]$ for all $t \geq 0$, respectively.

For time-varying parameter uncertainties, their variation rates Δp are not vanishing. Uncertainties of these quantities can be modeled by the intervals $\Delta p_k \in \left[\underline{\Delta p}_k \; ; \; \overline{\Delta p}_k\right]$ and $\Delta p\left(t\right) \in \left[\underline{\Delta p}\left(t\right) \; ; \; \overline{\Delta p}\left(t\right)\right]$, respectively. In Fig. 1, the influence of non-vanishing parameter variation rates Δp is depicted for the scalar case.

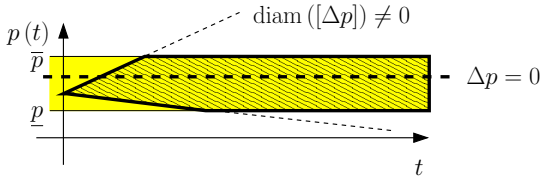


Figure 1. Time behavior of time-varying parameter uncertainties.

3 Steady State Analysis for Time-Invariant Systems

In order to analyze the steady state of a discrete-time dynamical system, the algebraic equations

$$x = g\left(x, p, u\right) \tag{3}$$

have to be solved. In the continuous-time case, the steady state is determined by solving the nonlinear algebraic equations

$$0 = f\left(x, p, u\right) \quad . \tag{4}$$

In both scenarios, interval enclosures for all physically relevant solutions $x = x\left(p, u\right)$ with $p \in [p]$ have to be determined. To deal with this problem, possible interval arithmetic approaches are:

Solution Approach 1
(i) Subdivision of the physically relevant domain $[x]$.
(ii) Consistency tests for all subintervals of $[x]$ by interval evaluation of g and f in (3) and (4).

Solution Approach 2
(i) Subdivision of physically relevant domain $[x]$ (*optional*).
(ii) Application of interval Newton methods, e.g. Krawczyk operator.

3.1 Rocker Arm

As a first application in steady state analysis of nonlinear systems with uncertainties, the tolerances of motion of the valve lifter depicted in Fig. 2 are determined for the known uncertain system parameters p_i, $i = 1, \ldots, 8$,

$$\begin{aligned} p_1 &:= R_G \;\; \in [16 \; ; \; 16.01] \\ p_2 &:= R_S \;\; \in [25 \; ; \; 25.01] \\ p_3 &:= Dist \in [42 \; ; \; 42.01] \\ p_4 &:= L_1 \;\; \in [27.5 \; ; \; 27.501] \\ p_5 &:= \Delta\epsilon \;\; \in [0.7 \; ; \; 0.7] \\ p_6 &:= K_2 \;\; \in [4 \; ; \; 4.001] \\ p_7 &:= L_4 \;\; \in [34 \; ; \; 34.01] \\ p_8 &:= R_1 \;\; \in [10 \; ; \; 10.001] \quad . \end{aligned} \tag{5}$$

The motion of the considered valve lifter is described by

$$\begin{aligned} x_1 := S_V(p_i) &= p_8 \; + \; p_6 \; - \; p_7 \cos\delta(p_i) \\ x_2 := V_V(p_i) &= p_7 H_4(p_i) \sin\delta(p_i) \\ x_3 := B_V(p_i) &= p_7 \cdot [H_4^2(p_i) \cos\delta(p_i) \\ &\quad + H_6(p_i) \sin\delta(p_i)] \quad , \end{aligned} \tag{6}$$

where the functions H_4, H_6, and δ are explicitly given by the geometry of the system, see [22]. In equation (6), the variable S_V denotes the position of the valve lifter, V_V its velocity, and B_V its acceleration.

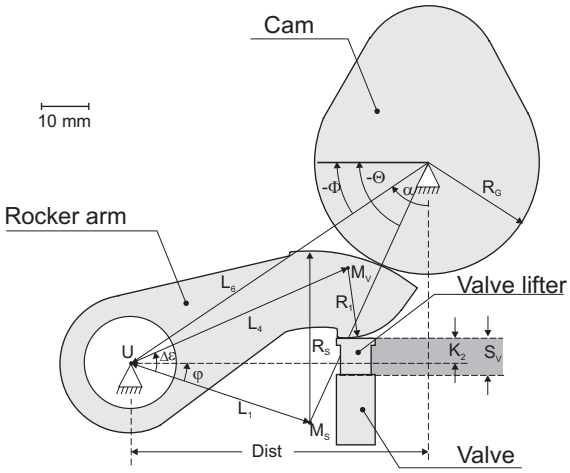


Figure 2. System parameters of a rocker arm.

The goal of the steady state analysis for this system is to determine guaranteed bounds for all variables x_j, $j = 1, 2, 3$ according to

$$\begin{aligned} x_1 \in [\underline{x}_1\,;\,\overline{x}_1] &= [\min S_V(p_i)\,;\max S_V(p_i)] \\ x_2 \in [\underline{x}_2\,;\,\overline{x}_2] &= [\min V_V(p_i)\,;\max V_V(p_i)] \\ x_3 \in [\underline{x}_3\,;\,\overline{x}_3] &= [\min B_V(p_i)\,;\max B_V(p_i)]\ . \end{aligned} \tag{7}$$

In the following, the results obtained by natural interval arithmetic, mean-value rule evaluation, and optimized interval arithmetic based on global optimization [3] including mean-value rule evaluation and monotonicity tests are summarized.

Natural interval evaluation

$$\begin{aligned} S_V(\Phi) &= [2.23940; 2.35208] \\ V_V(\Phi) &= [2.04417; 2.61488] \\ B_V(\Phi) &= [-6.23038; 10.29386] \end{aligned} \tag{8}$$

Mean-value rule evaluation

$$\begin{aligned} S_V(\Phi) &= [2.29515; 2.29634] \\ V_V(\Phi) &= [2.32611; 2.32874] \\ B_V(\Phi) &= [2.35106; 2.41480] \end{aligned} \tag{9}$$

For the optimized interval evaluation the following outer interval enclosures and inner interval enclosures have been determined:

Outer interval enclosures in optimized evaluation

$$\begin{aligned} S_V(\Phi) &= [\underline{2.295}244; \underline{2.296}247] \\ V_V(\Phi) &= [\underline{2.326}920; \underline{2.327}928] \\ B_V(\Phi) &= [\underline{2.382}394; \underline{2.383}468] \end{aligned} \tag{10}$$

Inner interval enclosures in optimized evaluation

$$\begin{aligned} S_V(\Phi) &= [\underline{2.295}318; \underline{2.296}160] \\ V_V(\Phi) &= [\underline{2.327}011; \underline{2.327}837] \\ B_V(\Phi) &= [\underline{2.382}495; \underline{2.383}367] \end{aligned} \tag{11}$$

The desired accuracy between the outer and inner interval bounds has been chosen as $\epsilon = 10^{-4}$ for each x_j. As displayed in the sketch in Fig. 3, the inner interval enclosures are always completely included in the outer ones. By the outer and inner interval bounds (10) and (11), an enclosure of the *true range* of the variable x_j is given. For the sake of comparison with non-validated evaluation techniques, the range of all x_j has been approximated by a Monte-Carlo simulation [2] using 10,000 samples. The resulting bounds are given in (12) and Fig. 4.

Monte-Carlo simulation

$$\begin{aligned} S_V(\Phi) &= [2.29529; 2.29621] \\ V_V(\Phi) &= [2.32797; 2.32789] \\ B_V(\Phi) &= [2.38248; 2.38339] \end{aligned} \tag{12}$$

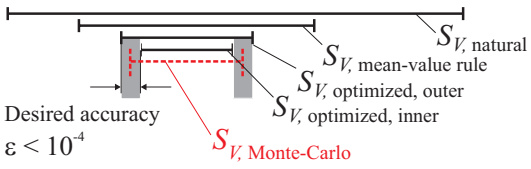


Figure 3. Reduction of overestimation by sophisticated interval techniques for range computation.

It should be pointed out that Monte-Carlo methods — especially for complex, higher-dimensional systems — can only provide tight bounds for the desired range if huge numbers of sampling points are used. Hence, it cannot be guaranteed that the bounds computed by Monte-Carlo simulations are contained within the inner and outer enclosures determined using interval arithmetic.

3.2 Micro Relay

As a second application, the estimation of system parameters for the micro relay displayed in Fig. 5 is considered [6]. Based on rough a priori enclosures of the range

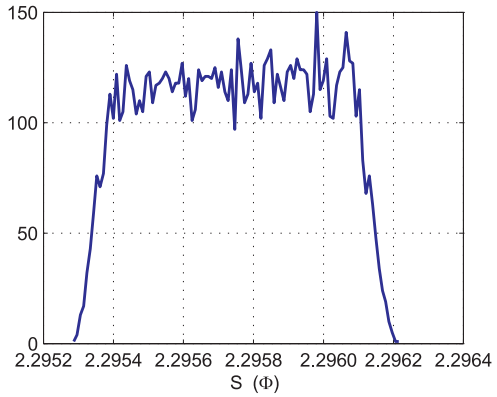


Figure 4. Monte-Carlo simulation as reference for the quality of the computed interval bounds of S_V.

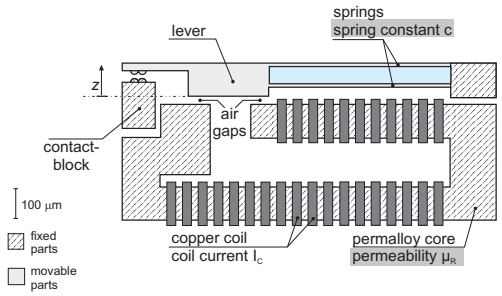


Figure 5. Micro relay.

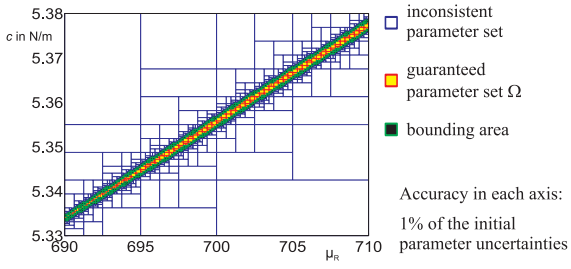


Figure 6. Consistent and inconsistent parameter sets for the micro relay within the a priori enclosures.

of the spring constant $c \in [5.33; 5.38]\,\mathrm{N/m}$ and the permeability $\mu_R \in [690; 710]$ of the permalloy core as well as uncertain measurements of the displacement z of the lever, the parameter estimates have to be improved such that only values are obtained which are consistent with all measured data. The measurement of the displacement has been carried out 23 times for various coil currents I_c with an uncertainty of $\pm 0.3\mu\mathrm{m}$ of the measured position. Using the measurement equations

$$\begin{aligned} z_j(\mu_R, c) &= \frac{2\gamma(\mu_R)}{3} \\ &\quad \cdot \left(\cos\left(\frac{1}{3} \arccos\left(1 - \frac{27\vartheta(c, I_{cj})}{2\gamma(\mu_R)^3} \right) \right) - 1 \right) \\ \gamma(\mu_R) &= \frac{L_{Fe} A_G}{2\mu_R A_{Fe}} + \delta_0 \\ \vartheta(c, I_{cj}) &= \frac{N^2 A_G \mu_0 I_{cj}^2}{4c}\,, \quad j = 1, \ldots, 23 \end{aligned} \tag{13}$$

the admissible set Ω of the spring constant c and the permeability μ_R is given by

$$\Omega = \left\{ \begin{bmatrix} \mu_R \\ c \end{bmatrix} \middle| \begin{array}{r} |\hat{z}_j(I_{cj}) - z(\mu_R, c)| \leq 0.3\mu\mathrm{m}, \\ j = 1, \ldots, 23 \end{array} \right\} . \tag{14}$$

The resulting guaranteed parameter set Ω and the bounding area containing the intervals which separate the consistent and inconsistent parameter values are depicted in Fig. 6. The accuracy of the computed sets can be influenced by specification of the maximum admissible width of each component of the undecided intervals. Here, an accuracy of 1% of the initial parameter uncertainties has been specified.

4 Discrete-Time Dynamical Systems with Time-Invariant Parameter Uncertainties

After the discussion of techniques and applications of steady state analysis of dynamical systems with uncertainties, the *dynamics* of discrete-time models and continuous-time models is analyzed in Sections 4 and 5, respectively.

According to Section 2, a discrete-time system is given by the state-space representation

$$\begin{bmatrix} x_{k+1} \\ p_{k+1} \end{bmatrix} = \begin{bmatrix} g_k\left(x_k, p_k, u_k, k\right) \\ g_{p,k}\left(p_k, \Delta p_k\right) \end{bmatrix} . \tag{15}$$

For time-invariant uncertainties, the relations $\Delta p_k = 0$ and $p_{k+1} = p_k$ hold for all $k \geq 0$. For the following analysis it is assumed that u_k is either a given open-loop or closed-loop control law. Using interval arithmetic evaluation of the mathematical system models for all uncertain parameters, guaranteed state enclosures have to be determined for each time step k for a given finite time horizon. In the following list of possible solution approaches, only those are

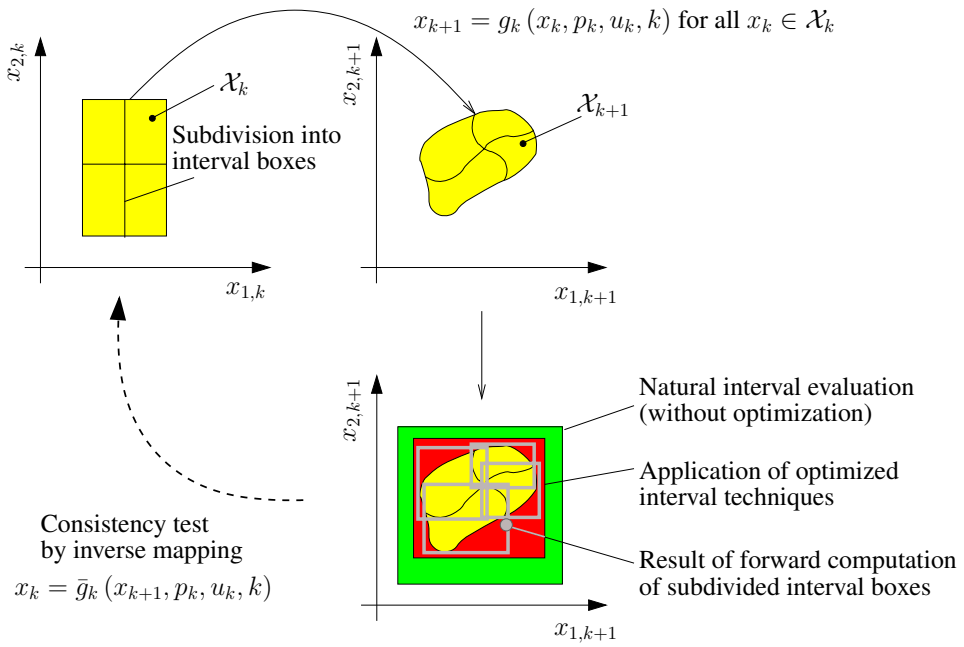


Figure 7. Interval arithmetic simulation of dynamical processes with subdivision into interval boxes, propagation of subintervals, and consistency tests.

mentioned which have been used in the selected applications [4].

Solution Approach 1. The basic approach for calculation of guaranteed state enclosures of discrete-time systems is the recursive computation of the state intervals

$$[x_{k+1}] = g_k\big([x_k]\,,[p_k]\,,u_k,k\big) \tag{16}$$

for open-loop control and

$$[x_{k+1}] = g_k\big([x_k]\,,[p_k]\,,u_k\,([x_k])\,,k\big) \tag{17}$$

in the case of closed-loop control.

Solution Approach 2. To reduce overestimation caused by the wrapping effect, computation of the state enclosures $[x_{k+1}]$ can be improved by coordinate transformations. The simplest possibility is a linear transformation according to

$$\begin{aligned} [x_k] &= T_k \cdot [\tilde{x}_k] \\ [\tilde{x}_{k+1}] &= T_{k+1}^{-1} \cdot g_k\big(T_k \cdot [\tilde{x}_k]\,,\big[p_k\big],u_k,k\big) \quad . \end{aligned} \tag{18}$$

Solution Approach 3. In Fig. 7, subdivision of state intervals $[x_{k+1}]$ is used for the computation of tight enclosures of complexly shaped regions if the coordinate transformation in the *Solution Approach 2* does not result in the desired quality.
(i) Consistency tests by inverse mapping of the state equation according to

$$[x_k] = \bar{g}_k\big([x_{k+1}]\,,[p_k]\,,u_k,k\big) \quad , \tag{19}$$

where the interval $[x_{k+1}]$ denotes the subintervals obtained by forward computation [10].
(ii) Interval Newton methods for state equations where the inverse mapping cannot be calculated analytically.
(iii) Merging of subintervals in case of small overestimation of the union of the merged subintervals is described in detail in [19].

Solution Approach 4. Computation of state variables $[x_{k+1}]$ by explicit replacement of $[x_k]$, $[x_{k-1}]$, …, $[x_2]$, $[x_1]$ in terms of the initial state $[x_0]$ and all parameter uncertainties $[p_0]$, $[p_1]$, …, $[p_{k-1}]$, $[p_k]$, i.e.,

$$\begin{aligned} [x_{k+1}] = g_k\Bigg(g_{k-1}\Big(\,\dots \\ g_1\Big(g_0\,([x_0]\,,[p_0]\,,u_0,0)\,,[p_1]\,,u_1,1\Big)\,\dots\Big)\Bigg) \quad . \end{aligned} \tag{20}$$

Here, mean-value rule evaluation, monotonicity tests, and global optimization techniques are useful approaches to significantly reduce overestimation caused by multiple dependency of the state equation (15) upon the components of the interval vectors $[x_k]$ and $[p_k]$.

4.1 Airbus Elevator

For the elevator control loop depicted in Fig. 8, interval enclosures for the actual elevator angle δ and the servo valve position x_V are of interest for a given time horizon [5].

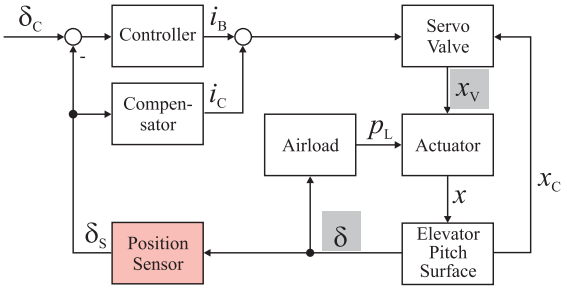


Figure 8. Airbus elevator.

This dynamical system can be modeled with sufficient accuracy by the discrete-time equations

$$
\begin{aligned}
x_{k+1} &= x_k + \sqrt{5}\cdot 10^5 \frac{B}{A} x_v h \Delta t,\\
y_k &= \delta = k_{SPAP}\, x_k,\\
h &= \boxed{\operatorname{sign}(z)\,\sqrt{|z|}},\\
z &= \left(\frac{2}{1+\frac{k^*_{MQ} B^2 x_V^2}{2\,A^2}} - 1 \right) (\Delta p_s - p_L \boxed{\operatorname{sign}(x_V)}),\\
p_L &= -\frac{100}{A}(a + b\,\delta)\, c\, v_{CAS}^2, \qquad (21)\\
x_V &= k_{SV}\, k_{SC}(i_C + i_B) - x_C,\\
x_C &= k_{Fb}\, k_C\, x,\\
i_B &= k_R\,(\delta_C - \delta_S),\\
i_C &= k_{Cp}\,\delta_S,\\
\delta_S &= r\,\delta + \delta_{offs}\ ,
\end{aligned}
$$

where the reference elevator angle is denoted by δ_C, the measured elevator angle by δ_S, the control output by i_B, the compensating current by i_C, the load pressure by p_L the hydraulic cylinder position by x, and the mechanical feedback by x_C. The functions highlighted by gray boxes in (21) are not continuously differentiable. The uncertainties of the position sensor in the closed-loop control are

$$r = [0.98\ ;\ 1.02] \quad \text{and} \quad \delta_{offs} = [-0.6\ ;\ 0.6]\ . \qquad (22)$$

For the desired accuracies $\epsilon_\delta = 0.1^\circ$ of the actual elevator angle δ and $\epsilon_x = 0.01\text{mm}$ of the servo valve position x_V, interval enclosures of their time responses are shown in Figs. 9 and 10, respectively.

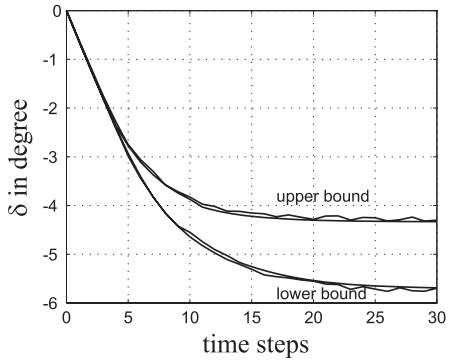


Figure 9. Interval bounds for the time response of the actual elevator angle δ.

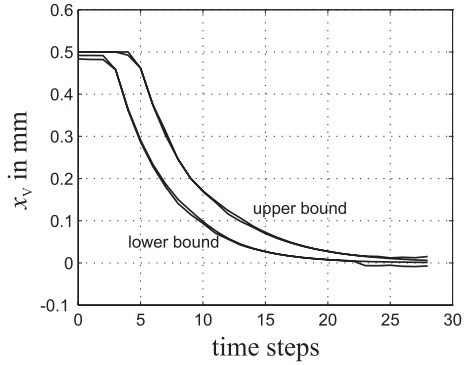


Figure 10. Interval bounds for the time response of the servo valve position x_V.

4.2 Common-Rail Injection System

Analogously to the previous application, the sensitivity of the closed-loop control of the common-rail injection system in Fig. 11 is analyzed [21]. This system is described by the discrete-time model given by the equations (23)–(26). The reference pressure is denoted by p_{ref}, the drive voltage by u, the tappet displacement by x_c, and the measured rail pressure by p_{sensor}.

Mathematical model of the common-rail injection system

Common-rail injection

$$
\begin{aligned}
p_{rail,k+1} &= f_{lim,p}\left(p_{rail,k} + \frac{q_{pump} - q_{ab1} - q_{ab2}}{V\,\kappa}\,\Delta t\right) \\
q_{ab1} &= A_{flow}\alpha_{furb}\sqrt{\frac{p_{rail,k} - p_{ab}}{5\,\rho}}, \quad \rho = \frac{824 - 0.68\,(t_\rho - 15)}{1 - \frac{0.06\,p_{rail,k}}{639\ + p_{rail,k}}} \\
A_{flow} &= \min\left(\frac{\pi}{2}\,x_{C,k}\left(d + \frac{x_{C,k}}{\sqrt{2}}\right),\ \frac{\pi}{16}\,d^2\right)
\end{aligned}
\tag{23}
$$

Magnetic valve

$$
\begin{aligned}
x_{C,k+1} &= f_{lim,x}(x_{C,k} + (F_{err} - c\,x_{C,k})\,\Delta t), \quad F_{err} = F_{hyd} - F_0 - F_{mag} \\
F_{mag} &= k_1\left(\omega\frac{f_{lim,z}\left(1 - \mathrm{e}^{-\frac{i_{L,k}}{k_3}}\right)}{f_{lim,n}(k_2 + 0.001\,x_{C,k})}\right)^2 \\
F_{hyd} &= k_0\,\frac{p_{rail,k} - p_{ab} + k_1\,x_{C,k}}{k_2 + x_{C,k}} \\
i_{L,k+1} &= i_{L,k} + \frac{u_{in} - R\,i_{L,k}}{L}\Delta t, \quad u_{in} = u_{batt}\,f_{lim,u}(u)
\end{aligned}
\tag{24}
$$

Controller

$$
\begin{aligned}
u &= f_{stat}(p_{ref}) + u_P + u_{I,k} + u_D \\
u_p &= K_R\,e_k, \quad e_k = p_{ref} - p_{sensor} \\
u_{I,k+1} &= u_{I,k} + e_k\,\frac{K_R}{T_I}\,f_{switch}(u_P, u_D) \\
u_D &= K_R T_D\,\frac{e_k - e_{k-1}}{\Delta t}
\end{aligned}
\tag{25}
$$

Sensor characteristic

$$
p_{sensor} = r\,p_{rail,k} + p_{offs} \tag{26}
$$

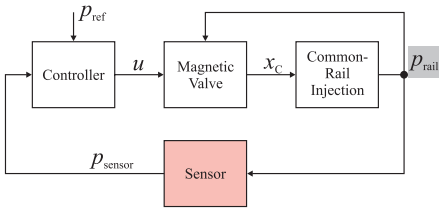


Figure 11. Common-rail injection system.

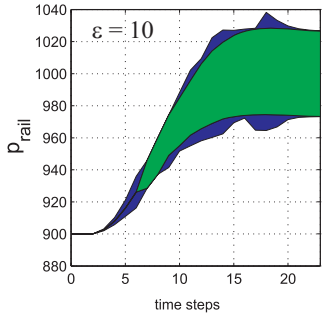


Figure 12. Inner and outer interval enclosures for the rail pressure p_{rail}.

The functions highlighted by gray boxes in (23)–(26) contain system-dependent static nonlinearities as well as saturation and switching characteristics which are not continuously differentiable. Hence, special treatment of these terms is necessary for the application of evaluation techniques which use partial derivatives of the state equations w.r.t. parameters and states. These are, for example, evaluation techniques aiming at the reduction of overestimation such as mean-value rule evaluation and monotonicity tests which are summarized in [4]. In the sensitivity analysis, the uncertainty $r \in [0.97\ ;\ 1.03]$ of the parameter of the pressure sensor is considered. The resulting interval enclosures for the actual rail pressure p_{rail} are shown in Fig. 12. The desired accuracy of the actual rail pressure p_{rail} in the global optimization approach used for computation of inner and outer interval bounds has been set to $\epsilon = 10$.

5 Continuous-Time Dynamical Systems with Time-Varying Parameter Uncertainties

5.1 Theoretical Background

In addition to discrete-time processes which have been discussed in the previous Section, continuous-time systems

described by sets of ordinary differential equations ODEs are widely used system representations in engineering. The considered ODEs are assumed to be given in state-space representation according to

$$\begin{bmatrix} \dot{x}_s(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} f_s(x_s(t), p(t), u(t), t) \\ \Delta p(t) \end{bmatrix} , \quad (27)$$

where the system parameters are time-varying, i.e., $\Delta p(t) \neq 0$ and $\mathrm{diam}([\Delta p]) \neq 0$ usually hold for all $t \geq 0$. For given bounded uncertainties of the initial state vector, given bounded parameter uncertainties, and given open-loop and closed-loop control laws $u(t)$ and $u(x(t))$, guaranteed state enclosures have to be determined at each point of time t for a given finite time horizon.

To simplify the notation for the solution approaches discussed in the following, the extended state vector

$$x(t) := \left[x_s^T(t) \quad p^T(t)\right]^T \quad (28)$$

is introduced. This allows to rewrite the state equations (27) in the form

$$f(\cdot) := \begin{bmatrix} f_s(x(t), u(t), t) \\ \Delta p(t) \end{bmatrix} . \quad (29)$$

Solution Approach 1. The computation of state enclosures of an initial value problem IVP by series expansion *with respect to time* according to

$$[x(t_{k+1})] = [x(t_k)] + h_k \cdot \phi([x(t_k)], u_k, t_k) + [e_k] \quad (30)$$

with

$$\phi(\cdot) := \sum_{i=1}^{\nu} \frac{h_k^{i-1}}{i!} \cdot \frac{d^{i-1} f(\cdot)}{dt^{i-1}} \quad (31)$$

and the discretization error interval

$$[e_k] := \left. \frac{h_k^{\nu+1}}{(\nu+1)!} \cdot \frac{d^{\nu} f(\cdot)}{dt^{\nu}} \right|_{[\tau_k],[B_k]} \quad (32)$$

is the basis for several validated ODE solvers such as VNODE by Nedialkov [14]. Here, the bounds for the discretization error have to be evaluated for a bounding box of all states and parameters $[B_k]$ which can be reached in the time interval $[\tau_k] := [t_k \ ; \ t_{k+1}]$. This bounding box is usually either determined with the help of the Picard iteration or with the help of higher order enclosure methods [15].

Solution Approach 2. Furthermore, state enclosures of IVPs can also be computed by series expansions with respect to time *and initial states*. This approach is implemented in the Taylor-model-based solver COSY VI by Berz and Makino [12].

Solution Approach 3. A novel approach for the computation of state enclosures $[x_{encl}(t)] := x_{app}(t) + [R(t)]$ of IVPs — not based on series expansions — relies on non-validated approximate solutions $x_{app}(t)$ and guaranteed error bounds $[R(t)]$. This technique is implemented in VALENCIA-IVP by Rauh and Auer [1].

5.2 Biological Wastewater Treatment

The dynamical behavior of biological wastewater treatment plants has to be robust w.r.t. changes of most system parameters [10, 19]. Furthermore, cost-effective plant operation demands for a reduction of the oxygen input rate into the aeration tank to its lowest possible value. Hence, it is necessary to find a suitable compromise between both prerequisites. To deal with this problem, interval arithmetic simulation of mathematical system models such as the Activated Sludge Model No. 1 ASM1 of the International Water Association, under consideration of time-varying uncertain system parameters, is a useful technique. In the following, the *Solution Approach 1* using a validated explicit Euler method with subdividing and merging of interval boxes is applied to simulation of a subsystem model of a wastewater treatment plant as shown in Fig. 13.

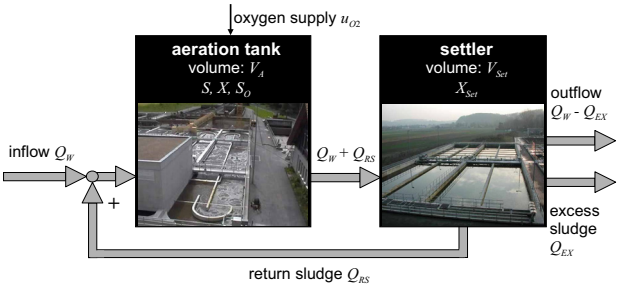


Figure 13. Biological wastewater treatment plant.

The state equations

$$\begin{aligned} \dot{S} =& \frac{Q_W}{V_A}(S_W - S) - \mu(S, S_O)\frac{1}{Y}X \\ \dot{X} =& -\frac{Q_W}{V_A}X + \frac{Q_{RS}}{V_A}(X_{Set} - X) \\ & + (\mu(S, S_O) - b)X \\ \dot{S}_O =& \frac{Q_W}{V_A}(S_{OW} - S_O) - \mu(S, S_O)\frac{1-Y}{Y}X \\ & + \frac{\rho_{O2}}{V_A}\left(1 - \frac{S_O}{S_{O,sat}}\right)u_{O2} \\ \dot{X}_{Set} =& \frac{Q_W + Q_{RS}}{V_{Set}}X - \frac{Q_{EX} + Q_{RS}}{V_{Set}}X_{Set} , \end{aligned} \quad (33)$$

where the nonlinear growth rate of substrate consuming bacteria is modeled by the Monod kinetics

$$\mu(S, S_O) = \hat{\mu}_H \frac{S}{S + K_S}\frac{S_O}{S_O + K_{OS}} , \quad (34)$$

describe the reduction of biodegradable organic substrate by heterotrophic bacteria. The state variables are the con-

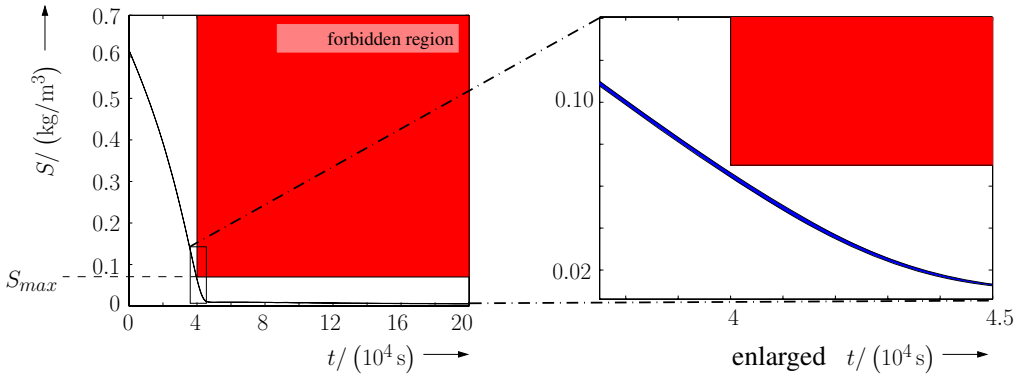


Figure 14. Substrate concentration S in the aeration tank.

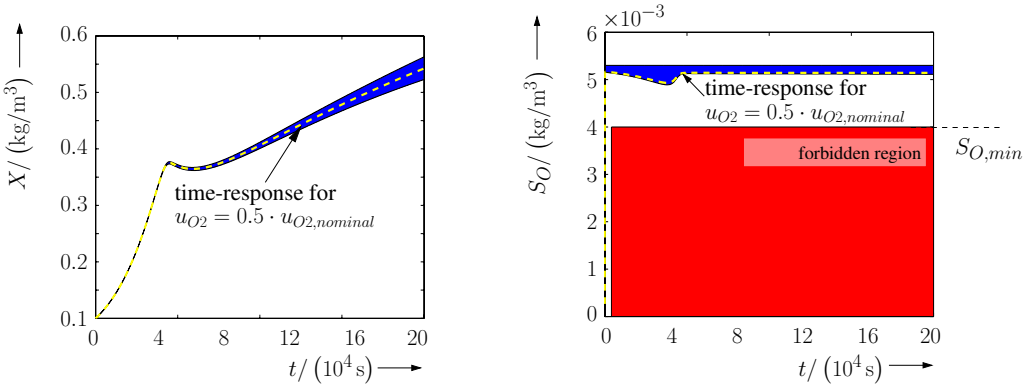


Figure 15. Bacteria concentration X and concentration S_O of dissolved oxygen.

centration S of organic substrate, the concentration X of bacteria, and the concentration S_O of dissolved oxygen in the aeration tank as well as the bacteria concentration X_{Set} in the settler.

With the prescribed bounds

$$\begin{aligned} \text{substrate:} &\begin{cases} S \text{ unbounded for} & t < 40,000\,\mathrm{s} \\ S \le S_{max} \text{ for} & t \ge 40,000\,\mathrm{s} \end{cases} \\ \text{oxygen:} &\begin{cases} S_O \text{ unbounded for} & t < 400\,\mathrm{s} \\ S_O \ge S_{O,min} \text{ for} & t \ge 400\,\mathrm{s} \end{cases} \end{aligned} \tag{35}$$

as robustness requirements, the computed validated state enclosures show that oxygen input rates from the interval $u_{O2} = [0.5\ ;\ 1.0] \cdot u_{O2,nominal}$ can be chosen without violation of the given bounds (35). Thus, the optimal choice of a constant oxygen input rate w.r.t. minimization of the operating costs is $u_{O2} = 0.5 \cdot u_{O2,nominal}$, see Figs. 14 and 15.

Note that this numerical proof of admissibility of control strategies for reliable plant operation using validated ODE solvers can be carried out analogously for control laws with time-varying oxygen input rates and for investigating parameter uncertainties. For details see [17].

6 State and Parameter Estimation Using Interval Observers

The applications which have been described in the previous Sections of this paper either dealt with estimation of system parameters in steady state or with the dynamical simulation of discrete-time and continuous-time systems without including any measured data. In the following, the concept of an interval observer [9] is discussed which relies on state-space representations of discrete-time systems as in eq. (1) and continuous-time systems as in eq. (2).

In addition to the system dynamics, mathematical models of the measurement process are necessary. They are

given by

$$y_{k+1} = h_{k+1}\left(x_{k+1}, p_{k+1}, \delta_{k+1}, u_{k+1}, k+1\right) \tag{36}$$

and

$$y\left(t_{k+1}\right) = h\left(x\left(t\right), p\left(t\right), \delta\left(t\right), u\left(t\right), t\right)\Big|_{t=t_{k+1}} \tag{37}$$

for discrete-time and continuous-time processes, respectively. In both cases, it is assumed that new measured data only become available at discrete points of time. For the sake of simplicity, the vector p of parameter uncertainties is now redefined such that it consists of the parameters of both the dynamical system model and the measurement model. Using the measurement equations (36) and (37), a model-based reconstruction of the state vector x as well as the parameter vector p is performed under consideration of the bounded measurement uncertainties $\delta \in [\delta] := \left[\underline{\delta}\,;\,\overline{\delta}\right]$.

The block diagram of the interval observer in Fig. 16 shows the two basic steps of state estimation. In the *prediction step*, propagation of all uncertainties is performed with the help of the mathematical model of the system dynamics until the point of time at which measured data are available. Then, the *correction step* eliminates those parts of the state enclosures (obtained by the prediction) which are inconsistent with the model of the measurement process under consideration of its uncertainties.

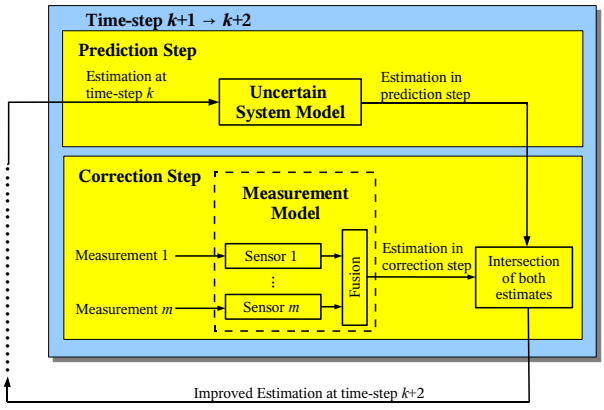


Figure 16. Interval observer.

In the following applications, state and parameter estimates computed by the interval observer are presented.

6.1 Electrostatic Micro Actuator

First, the electrostatic micro actuator in Fig. 17 is considered. For this device, the not directly measured initial gap x_{20} between the two plates of the capacitor, the position

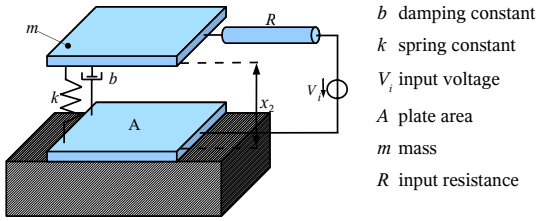


Figure 17. Electrostatic micro actuator.

$x_2\left(t\right)$, and the velocity $\dot{x}_2\left(t\right)$ have to be estimated. The dynamical system model is given by the ODEs [20]

$$\begin{aligned}
\dot{x}_1 &= \frac{1}{R}\left(V_i - \frac{x_1 x_2}{\epsilon A}\right) \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= \frac{-1}{m}\left(\frac{x_1^2}{2\epsilon A} + k(x_2 - x_{20}) + b x_3\right)
\end{aligned} \tag{38}$$

with the uncertain initial conditions

$$x(0) \in \begin{bmatrix}[x_{10}] \\ [x_{20}] \\ [x_{30}]\end{bmatrix} = \begin{bmatrix}[0;0] \\ [0.9;1.1] \\ [0;0]\end{bmatrix} \tag{39}$$

and the time-invariant uncertain spring constant $k \in [0.8;1.2]$. Using the measurement equations

$$y_1 = \frac{x_1 x_2}{\epsilon A} + \delta_1 \quad \text{and} \quad y_2 = \frac{x_2}{\epsilon A} + \delta_2 \tag{40}$$

with the uncertainties $\delta_1 \in [-3;3]\cdot 10^{-4}$ and $\delta_2 \in [-1;1]\cdot 10^{-4}$, the estimates in Fig. 18 are obtained. A significant reduction of the initial uncertainty $[x_{20}]$ by the model-based state and parameter estimation approach is obvious.

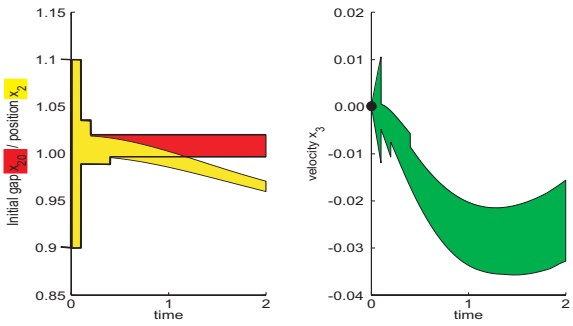


Figure 18. Estimates of position and velocity of the electrostatic micro actuator.

6.2 Micropositioning System

As a second application for the interval observer, the micropositioning system shown in Fig. 19 is discussed.

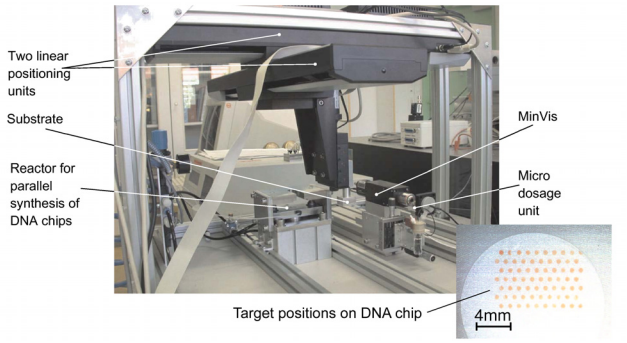


Figure 19. Micropositioning system.

Here, the task is the guaranteed positioning of microdrops on a DNA chip within given tolerances. For that purpose, the positioning unit is described by dynamical second order models in each axis. The dominating uncertainties in this system are the static friction coefficient F_s as well as the sliding friction coefficient μ which are depicted in the friction characteristic in Fig. 20.

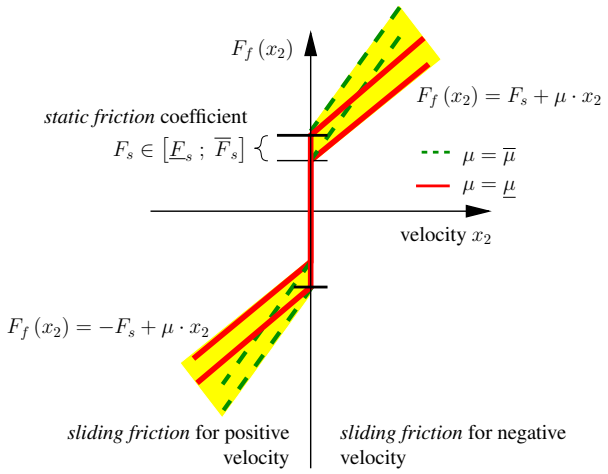


Figure 20. Friction characteristic.

The algorithm which is used for model-based estimation of both position and velocity of the positioning unit relies on validated integration of the dynamical system model in the prediction step. Here, all points of time have to be detected at which switchings between sliding and static friction occur. The exact procedure using a state transition diagram which is evaluated for all interval uncertainties has been published in [18].

In Fig. 21, results for two different cases are illustrated:
(i) The black solid lines show the influence of the parameter uncertainties in the friction characteristic in terms of worst-case bounds of the system's position.
(ii) Additionally, uncertain measurement information of the position is considered (marked by black dots). Then, compared to case (i), the bounds of the position estimate can be reduced significantly.

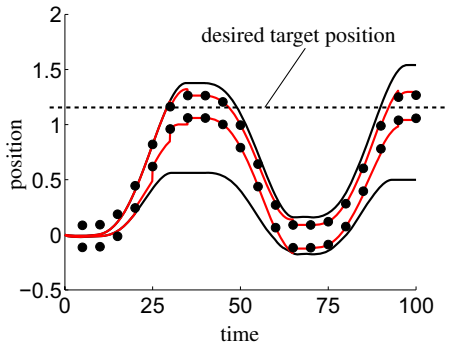


Figure 21. Position estimates.

7 Conclusions

The applications presented in this paper have been chosen to demonstrate how interval arithmetic evaluation methods for steady state analysis as well as simulation of both discrete-time and continuous-time systems can be applied successfully in engineering. The common goal of all applications has been to compute guaranteed enclosures of all physically relevant states and parameters. Besides simulation of the dynamical system behavior in open-loop and closed-loop operation, the design of a model-based interval observer for guaranteed state and parameter estimation has been described. In this approach, measurement information can be used efficiently to eliminate parts of the state enclosures which are inconsistent either with the mathematical model of the system dynamics or with the model of the measurement process.

As shown in this contribution, the most important property of interval algorithms, namely the ability to compute guaranteed enclosures [7], is especially relevant in the analysis and the design of technical and non-technical systems. Based on mathematical system models, the worst-case influence of uncertainties as well as the robustness, reliability, cost-effectiveness, and safety of a system can be investigated by computation of guaranteed bounds of the corresponding system states.

In order to apply interval algorithms to a wider class of problems, further general implementations have to be made

available. Additionally, future research aiming at the development of improved and novel interval arithmetic tools has to consider the necessity to deal with uncertain dynamical systems which include discontinuities and model switchings in the systems' representations, e.g. hystereses in electro-mechanical applications. Finally, the development of interval arithmetic methods for the computation of robust and optimal control strategies [11, 16] for high-dimensional nonlinear systems with non-negligible influence of uncertainties will be an important field for future developments.

Future research will be directed towards both establishing interval techniques in the computing mainstream by further developing software tools and — most important — demonstrating successful applications.

References

[1] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SMARTMOBILE: Improvement of Performance by VALENCIA-IVP. In *Proc. of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice*, Lecture Notes in Computer Science, 2006. In print.

[2] J. M. Hammersley and D. C. Handscomb. *Monte-Carlo Methods*. John Wiley & Sons, New York, 1964.

[3] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[4] J. Heeks. *Charakterisierung unsicherer Systeme mit intervallarithmetischen Methoden*. PhD thesis, Universität Ulm, Abteilung Mess-, Regel- und Mikrotechnik, 2002.

[5] J. Heeks, E. P. Hofer, B. Tibken, K. Lunde, and K. Thorwart. Simulation of a Controlled Aircraft Elevator Under Sensor Uncertainties. In W. Krämer and J. W. von Gudenberg, editors, *Scientific Computing, Validated Numerics, Internal Methods*, Kluwer Academic/Plenum Publishers, New York, pages 227–237. 2001.

[6] E. P. Hofer, B. Tibken, and C. Rembe. Guaranteed Parameter Estimation for Characterization of Microdevices. In E. Reithmeier and G. Leitmann, editors, *Proc. of 10th Workshop on Dynamics and Control, Lambrecht, Germany, 1998*, Complex Dynamical Systems with Incomplete Information, pages 81–93. Shaker Verlag, Aachen, 1999.

[7] E. P. Hofer, B. Tibken, and M. Vlach. Traditional Parameter Estimation Versus Estimation of Guaranteed Parameter Sets. In W. Krämer and J. W. von Gudenberg, editors, *Scientific Computing, Validated Numerics, Internal Methods*, Kluwer Academic/Plenum Publishers, New York, pages 241–253. 2001.

[8] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter. *Applied Interval Analysis*. Springer, London, 2001.

[9] M. Kletting, A. Rauh, H. Aschemann, and E. P. Hofer. Interval Observer Design for Nonlinear Systems with Uncertain Time-Varying Parameters. In *Proc. of 12th IEEE Intl. Conference on Methods and Models in Automation and Robotics MMAR*, pages 361–366, Miedzyzdroje, Poland, 2006.

[10] M. Kletting, A. Rauh, H. Aschemann, and E. P. Hofer. Consistency Tests in Guaranteed Simulation of Nonlinear Uncertain Systems with Application to an Activated Sludge Process. *Journal of Computational and Applied Mathematics*, 199(2):213–219, 2007.

[11] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization for Dynamic Systems Using Interval Analysis. In *Book of Abstracts of 12th GAMM-IMACS Intl. Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006*, page 74, Duisburg, Germany, 2006.

[12] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, 1998.

[13] R. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[14] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 1999.

[15] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An Effective High-Order Interval Method for Validating Existence and Uniqueness of the Solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001.

[16] A. Rauh and E. P. Hofer. Interval Arithmetic Optimization Techniques for Uncertain Discrete-Time Systems. In E. P. Hofer and E. Reithmeier, editors, *Proc. of 13th Int. Workshop on Dynamics and Control, Wiesensteig, Germany, 2005*, Modeling and Control of Autonomous Decision Support Based Systems, pages 141–148. Shaker Verlag, Aachen, 2005.

[17] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Robust Controller Design for Bounded State and Control Variables and Uncertain Parameters Using Interval Methods. In *Proc. of 5th Intl. Conference on Control and Automation ICCA*, pages 777–782, Budapest, Hungary, 2005.

[18] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Interval Methods for Simulation of Dynamical Systems with State-Dependent Switching Characteristics. In *Proc. of IEEE Intl. Conference on Control Applications CCA 2006*, pages 355 – 360, Munich, Germany, 2006.

[19] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Reduction of Overestimation in Interval Arithmetic Simulation of Biological Wastewater Treatment Processes. *Journal of Computational and Applied Mathematics*, 199(2):207–212, 2007.

[20] A. Rauh, M. Kletting, and E. P. Hofer. Model-Based State and Parameter Estimation for Micro-Mechatronic Systems with Interval Bounded Uncertainties. In A. Weckenmann, editor, *CD-Proc. of 10th CIRP Intl. Conference on Computer Aided Tolerancing, Erlangen, Germany, 2007*, Reports from the Chair Quality Management and Manufacturing Metrology, QFM Report 16. Shaker Verlag, Aachen, 2007.

[21] B. Tibken, E. P. Hofer, J. Heeks, K. Thorwart, and H. Aschemann. Simulation of a Common Rail Fuel Injection System Using Interval Arithmetic. SCAN 2002. Paris, France, 2002.

[22] B. Tibken, E. P. Hofer, and W. Seibold. Quality Control of Valve Push Rods Using Interval Artihmetic. In *Proc. of 14th Triennial World Congress Intl. Federation of Automatic Control IFAC*, volume A, pages 409–412, Beijing, P. R. China, 1999.

Operator Dependant Compensated Algorithms

Philippe Langlois Nicolas Louvet
DALI at ELIAUS Laboratory, Université de Perpignan Via Domitia
52, avenue Paul Alduy, F-66860 Perpignan, France
[langlois, nicolas.louvet]@univ-perp.fr

Abstract

Compensated algorithms improve the accuracy of a result evaluating a correcting term that compensates the finite precision of the computation. The implementation core of compensated algorithms is the computation of the rounding errors generated by the floating point operators. We focus this operator dependency discussing how to manage and to benefit from floating point arithmetic implemented through a fused multiply and add operator. We consider the compensation of dot product and polynomial evaluation with Horner iteration. In each case we provide theoretical a priori error bounds and numerical experiments to exhibit the best algorithmic choices with respect to accuracy or performance issues.

1. Introduction

Different techniques and several softwares aim to improve the accuracy of results computed in a fixed finite precision, *e.g.*, in IEEE-754 floating point arithmetic [10].

A natural way to improve the accuracy of a given computation is to increase the working precision. For this purpose, numerous multiprecision libraries are available when the computing precision is not large enough to guarantee a prescribed accuracy [4, 2, 14]. The computing-time overhead of such arbitrarily precise computation limits its use to applications for which running-time is not crucial. When twice or four times the IEEE-754 double precision is sufficient, actual and effective solutions are double-double or quad-double libraries [8, 1]. For example a double-double number is an unevaluated sum of two IEEE-754 double precision numbers and its associated arithmetic provides at least 106 bits of significand. These fixed-length expansions are currently embedded in major developments such as for example within the new extended and mixed precision BLAS [12]. Such libraries benefit from good performances in term of running-time and also from a wide applicability since they provide extended precision for classic arithmetic operators and elementary functions.

Compensating a given algorithm is a less generic process than the simple plug-in of the extended precision facilities previously mentioned. Nevertheless farther presented results will exhibit that when available, compensated algorithms run always faster than the corresponding ones with extended precision libraries. Compensated algorithms implement the computation of a correcting term that approximates the errors generated by the finite precision evaluation of the algorithm. This computation relies on error-free transformations (EFT) as named in [16]. EFT are properties that describe the final forward error such that (an approximate of) this error can be computed only using the current working precision. Such EFT for arithmetic operators, dot product and polynomial evaluation will be given hereafter. The core of the EFT computation depends on low-level arithmetic properties (which is also the case for extended precision libraries); most of them are clearly defined by the IEEE-754 standard. Nevertheless new questions raise when IEEE-754 compliant add or multiply operators are implemented from a unique fused multiply and add instruction. The fused multiply and add instruction (FMA) is available on some current processors, such as the IBM Power PC or the Intel Itanium. Given a, b and c three floating point point values, this instruction computes the expression $a \times b + c$ with only one final rounding error [13].

The FMA can be used to improve algorithms based on error-free transformations in two ways. First, it allows us to compute the EFT for the product of two floating point values in a very efficient way: algorithm TwoProd recalled hereafter computes this EFT in only two flops when a FMA is available [15, 13]. On the other hand, an algorithm that computes an EFT for the FMA has been proposed in [3]. In particular, it is proved that the EFT for the FMA is the sum of three floating point numbers. Assuming an IEEE-754 like floating point arithmetic with the round to the nearest rounding mode, algorithm ThreeFMA

Table 1. Summary of algorithms

routine	description
HornerFMA	IEEE-754 double precision with FMA (Algorithm 5)
CompHornerFMA	Compensated HornerFMA (Algorithm 7)
CompHorner	Compensated Horner (Algorithm 9)
DDHorner	Horner algorithm performed with the double-double format + FMA
DotFMA	Dot product algorithm with FMA (Algorithm 11)
CompDotFMA	Compensated DotFMA (Algorithm 12)
CompDot	Compensated Dot (Algorithm 13)
DDDot	Dot product with the double-double format + FMA

computes three floating point numbers x, y and z such that

$$a \times b + c = x + y + z \quad \text{with} \quad x = \mathsf{FMA}\,(a, b, c)\,.$$

In this paper we focus on the FMA dependency of compensated algorithms. We discuss how to manage and to benefit from this fused multiply and add operator. Notations are presented in Section 2 and then error-free transformations are introduced in Section 3.

We consider the compensation of polynomial evaluation with Horner iteration and dot product, respectively in Section 4 and Section 5. In each case we provide theoretical *a priori* error bounds and numerical experiments to exhibit the best algorithmic choices with respect to accuracy or performance issues. *A priori* error bounds prove that the FMA does not significantly improve the worst-case error – even if implementations with FMA suffer from twice less rounding errors than without. Experiments also illustrate this similar behavior in terms of accuracy for both original and compensated algorithms.

Running-time issues are different for compensated algorithms. We conclude that FMA should be avoided in the main computation (replacing it by add or multiply operators) but preferred in the compensating process (namely to compute the error generated by the multiply operator). This should motivate further research to less costly algorithms or availability of low level primitives that compute the error generated by the FMA. Hence the whole computation of compensated algorithms would benefit from the fused multiply and add instruction.

2. Notations

Throughout the paper, we assume a floating point arithmetic adhering to the IEEE-754 floating point standard [10]. We constraint all the computations to be performed in one working precision, with the "round to the nearest" rounding mode. We also assume that no overflow nor underflow occurs during the computations. Next notations are standard (see [9, chap. 2] for example). $\mathbb{F}$ is the set of all normalized floating point numbers and $\mathbf{u}$ denotes the unit roundoff, that is half the spacing between 1 and the next larger representable floating point value. For IEEE-754 double precision with rounding to the nearest, we have $\mathbf{u} = 2^{-53} \approx 1.11 \cdot 10^{-16}$.

The symbols $\oplus$, $\ominus$ and $\otimes$ represent respectively the floating point addition, subtraction and multiplication. For more complex arithmetic expressions, $fl(\cdot)$ denotes the result of a floating point computation where every operation inside the parenthesis is performed in the working precision. So we have for example, $a \oplus b = fl(a + b)$.

When no underflow nor overflow occurs, the following standard model describes the accuracy of every considered floating point computation. For two floating point numbers a and b and for $\circ$ in $\{+, -, \times\}$, the floating point evaluation $fl(a \circ b)$ of $a \circ b$ is such that

$$fl(a \circ b) = (a \circ b)(1 + \varepsilon_1) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq \mathbf{u}. \quad (1)$$

To keep track of the $(1+\varepsilon)$ factors in next error analysis, we use the classic $(1 + \theta_k)$ and γ_k notations [9, chap. 3]. For any positive integer k, θ_k denotes a quantity bounded according to

$$|\theta_k| \leq \gamma_k = \frac{k\mathbf{u}}{1 - k\mathbf{u}}.$$

When using these notations, we always implicitly assume $k\mathbf{u} < 1$. In farther error analysis, we essentially use the following relations,

$$(1 + \theta_k)(1 + \theta_j) \leq (1 + \theta_{k+j}), \quad k\mathbf{u} \leq \gamma_k, \quad \gamma_k \leq \gamma_{k+1}.$$

3. Error Free Transformations (EFT)

First we review error free transformations (EFT) known for the elementary floating point operations $+$, $-$ and $\times$.

Let $\circ$ be an operator in $\{+, -, \times\}$, a and b be two floating point numbers, and $x = fl(a \circ b)$. Then it exists a floating point value y such that

$$a \circ b = x + y. \quad (2)$$

Table 2. Summary of *a priori* bounds and flop counts.

Algorithm	*A priori* bound for the relative accuracy	Number of flop
HornerFMA	$\gamma_n \operatorname{cond}(p,x)$	n
Horner	$\gamma_{2n} \operatorname{cond}(p,x)$	$2n$
CompHornerFMA	$\mathbf{u} + \gamma_n\gamma_{n+1} \operatorname{cond}(p,x)$	$19n$
CompHorner	$\mathbf{u} + \gamma_n\gamma_{2n+1} \operatorname{cond}(p,x)$	$10n-1$
DotFMA	$\gamma_n \operatorname{cond}(x^Ty)/2$	n
Dot	$\gamma_n \operatorname{cond}(x^Ty)/2$	$2n-1$
CompDotFMA	$\mathbf{u} + \mathbf{u}\gamma_{n+1} \operatorname{cond}(x^Ty)/2$	$19n-16$
CompDot	$\mathbf{u} + \gamma_n^2 \operatorname{cond}(x^Ty)/2$	$10n-7$

The difference y between the exact result and the computed result is the rounding error generated by the computation of x. Let us emphasize that relation (2) between four floating point values only relies on real operators and exact equality. Ogita *et al.* [16] name such a transformation an error free transformation (EFT). The practical interest of the EFT comes from next Algorithms 1 and 2 that compute the exact error term y for $\circ = +$ and $\circ = \times$.

For the EFT of the addition we use Algorithm 1, the well known TwoSum algorithm by Knuth [11] that requires 6 flop (floating point operations).

Usually, the well known algorithm TwoProd by Veltkamp and Dekker (see [5]) is used for the EFT of the product. TwoProd requires 17 floating point operations. Nevertheless, TwoProd can be rewritten very efficiently when a FMA is available. For a, b and c in $\mathbb{F}$, FMA (a,b,c) is the exact result $a \times b + c$ rounded to the nearest floating point value. Thus, $y = a \times b - a \otimes b =$ FMA $(a, b, -(a \otimes b))$, and TwoProd now only requires two flop.

The next theorem exhibits the previously announced properties of TwoSum and TwoProd.

Theorem 1 ([16]). *Let a, b in $\mathbb{F}$ and $x, y \in \mathbb{F}$ such that $[x,y] =$ TwoSum(a,b) (Algorithm 1). Then, even in the presence of underflow,*

$$a+b = x+y, \quad x = a \oplus b, \quad |y| \le \mathbf{u}|x|, \quad |y| \le \mathbf{u}|a+b|.$$

Let $a, b \in \mathbb{F}$ and $x, y \in \mathbb{F}$ such that $[x,y] =$ TwoProd(a,b) (Algorithm 2). Then, if no underflow occurs,

$$a \times b = x+y, \quad x = a \otimes b, \quad |y| \le \mathbf{u}|x|, \quad |y| \le \mathbf{u}|a \times b|.$$

An algorithm that computes an EFT for the FMA has been recently given by Boldo and Muller [3]. The EFT of a FMA operation cannot be represented as a sum of two floating point numbers, as it is the case for the addition and for the product. Therefore, the following algorithm ThreeFMA produces three floating point numbers. For efficiency reasons, we slightly modify the algorithm from [3] such that ThreeFMA here performs no renormalization of the final result. Algorithm 3 requires 17 flop. It satisfies the following properties.

Theorem 2 ([3]). *Given a, b, and c three floating point values, let x, y and z be the three floating point numbers such that $[x,y,z] =$ ThreeFMA (a,b,c). Then we have*

- $a \times b + c = x + y + z$ *exactly, with* $x =$ FMA (a,b,c),
- $|y+z| \le \mathbf{u}|x|$ *and* $|y+z| \le \mathbf{u}|a \times b + c|$,
- $y = 0$ *or* $|y| > |z|$.

We notice that the algorithms presented in this section only require well optimizable floating point operations. They do not use branches nor access to the mantissa that can be time-consuming.

Two error free transformations for polynomial evaluation are introduced in next Section 4. Relation (7) exhibits the exact rounding error generated by the Horner algorithm when its inner iteration uses a FMA; Relation (10) applies when no FMA appears in the Horner algorithm.

4. Polynomial evaluation

We consider the evaluation of $p(x) = \sum_{i=0}^{n} a_i x^i$, where the data x and the polynomial coefficients a_i are floating point numbers. We study the two versions of the classic Horner algorithm (without or with the FMA) and associated compensated Horner algorithms.

We recall that the classic condition number of the evaluation of $p(x)$ is

$$\operatorname{cond}(p,x) = \frac{\sum_{i=0}^{n} |a_i||x|^i}{|\sum_{i=0}^{n} a_i x^i|} = \frac{\widetilde{p}(x)}{|p(x)|}. \tag{3}$$

4.1. Horner algorithms

For any floating point value x, Horner (p,x) is the result of the floating point evaluation of the polynomial p at x using the Horner algorithm (Algorithm 4).

Algorithm 1. EFT of the sum of two floating point numbers.

function $[x, y] = \mathsf{TwoSum}\,(a, b)$
  $x = a \oplus b$
  $z = x \ominus a$
  $y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

Algorithm 2. EFT of the product of two floating point numbers with a FMA.

function $[x, y] = \mathsf{TwoProd}\,(a, b)$
  $x = a \otimes b$
  $y = \mathsf{FMA}\,(a, b, -x)$

Algorithm 3. EFT for the FMA operation.

function $[x, y, z] = \mathsf{ThreeFMA}\,(a, b, c)$
  $x = \mathsf{FMA}\,(a, b, c)$
  $(u_1, u_2) = \mathsf{TwoProd}\,(a, b)$
  $(\alpha_1, z) = \mathsf{TwoSum}\,(b, u_2)$
  $(\beta_1, \beta_2) = \mathsf{TwoSum}\,(u_1, \alpha_1)$
  $y = (\beta_1 \ominus x) \oplus \beta_2$

Algorithm 4. Horner algorithm

function $r_0 = \mathsf{Horner}\,(p, x)$
$r_n = a_n$
for $i = n - 1 : -1 : 0$
  $r_i = r_{i+1} \otimes x \oplus a_i$
end

A forward error bound for the result of Algorithm 4 is (see [9, p.95])

$$|p(x) - \mathsf{Horner}\,(p, x)\,| \leq \gamma_{2n}\,\widetilde{p}(x). \tag{4}$$

So, the accuracy of the computed evaluation is linked to the condition number of the polynomial evaluation as follows,

$$\frac{|p(x) - \mathsf{Horner}\,(p, x)\,|}{|p(x)|} \leq \gamma_{2n}\,\mathrm{cond}(p, x). \tag{5}$$

Clearly, the condition number (3) can be arbitrarily large. In particular, when $\mathrm{cond}(p, x) > \gamma_{2n}^{-1}$, we cannot guarantee that the computed result $\mathsf{Horner}\,(p, x)$ contains any correct digit.

If a FMA instruction is available on the considered architecture, then we can change the computation of $r_i = r_{i+1} \otimes x \oplus a_i$ in Algorithm 4 by $r_i = \mathsf{FMA}\,(r_{i+1}, x, a_i)$. This gives the following algorithm HornerFMA (Algorithm 5).

Algorithm 5. Horner algorithm with FMA

function $r_0 = \mathsf{HornerFMA}\,(p, x)$
$r_n = a_n$
for $i = n - 1 : -1 : 0$
  $r_i = \mathsf{FMA}\,(r_{i+1}, x, a_i)$
end

This slightly improves the error bound since we write now,

$$\frac{|p(x) - \mathsf{HornerFMA}\,(p, x)\,|}{|p(x)|} \leq \gamma_n\,\mathrm{cond}(p, x). \tag{6}$$

With the FMA, the number of floating point operations involved in the computation is also divided by two and so is the worst case error.

4.2. Compensating HornerFMA

As previously mentioned, next EFT for the polynomial evaluation with HornerFMA exhibits the exact rounding error generated by this algorithm. Following algorithm EFTHornerFMA computes this EFT thanks to ThreeFMA (Algorithm 3).

Algorithm 6. EFT for HornerFMA

function $[u_0, p_\varepsilon, p_\varphi] = \mathsf{EFTHornerFMA}(p, x)$
$u_n = a_n$
for $i = n - 1 : -1 : 0$
  $[u_i, \varepsilon_i, \varphi_i] = \mathsf{ThreeFMA}\,(u_{i+1}, x, a_i)$
  Let ε_i be the coefficient of degree i in p_ε
  Let φ_i be the coefficient of degree i in p_φ
end

Theorem 3. *Let $p(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial of degree n with floating point coefficients, and let x be a floating point value. Algorithm 6 computes both*

- *the floating point evaluation* $\mathsf{HornerFMA}\,(p, x)$ *(Algorithm 5), and*
- *two polynomials p_ϵ and p_φ, of degree $n-1$, with floating point coefficients;*

we write

$$[\mathsf{HornerFMA}\,(p, x)\,, p_\varepsilon, p_\varphi] = \mathsf{EFTHornerFMA}\,(p, x)\,.$$

Algorithm 6 requires $17n$ floating point operations.

We have the next EFT,

$$p(x) = \mathsf{HornerFMA}\,(p, x) + \widetilde{(p_\varepsilon + p_\varphi)}(x), \tag{7}$$

with

$$\widetilde{(p_\varepsilon + p_\varphi)}(x) \leq \gamma_n\,\widetilde{p}(x).$$

As before we have $\widetilde{(p_\varepsilon + p_\varphi)}(x) = \sum_{i=0}^{n-1} |\varepsilon_i + \varphi_i||x^i|$. Relation (7) means that EFTHornerFMA is an EFT for the polynomial evaluation with the Horner algorithm when the FMA is used. From this relation, the global forward error

affecting the floating point evaluation of p at x according to the Horner algorithm is

$$p(x) - \mathsf{HornerFMA}\,(p,x) = (p_\varepsilon + p_\varphi)(x), \qquad (8)$$

where the coefficients of the polynomials p_ε and p_φ are exactly computed by EFTHornerFMA (Algorithm 6), together with the approximate $\mathsf{HornerFMA}\,(p,x)$. Therefore, the key of the following compensated algorithm is to compute an approximate c of the global error (8) in working precision, and then to compute a corrected result

$$r = \mathsf{HornerFMA}\,(p,x) \oplus c.$$

We say that c is a correcting term for the initial result $\mathsf{HornerFMA}\,(p,x)$. The corrected result r is expected to be more accurate than $\mathsf{HornerFMA}\,(p,x)$ as proved in the sequel of the section. We compute the correcting term c by evaluating the polynomial whose coefficients are those of $p_\varepsilon + p_\varphi$ rounded to the nearest floating point value, *i.e.*, $c = \mathsf{HornerFMA}\,(p_\varepsilon + p_\sigma, x)$. We can now describe the compensated algorithm for polynomial evaluation.

Algorithm 7. Compensated HornerFMA

function $r = \mathsf{CompHornerFMA}\,(p,x)$
$[h, p_\varepsilon, p_\varphi] = \mathsf{EFTHornerFMA}\,(p,x)$
$c = \mathsf{HornerFMA}\,(p_\varepsilon \oplus p_\varphi, x)$
$r = h \oplus c$

We state hereafter that the result of a polynomial evaluation computed with Algorithm 7 is as accurate as if computed by the classic Horner algorithm using twice the working precision and then rounded to the working precision (proofs are detailed in [7]).

Theorem 4. *Given a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ of degree n with floating point coefficients, and x a floating point value. We consider the result* $\mathsf{CompHornerFMA}\,(p,x)$ *computed by Algorithm 7. Then,*

$$|\mathsf{CompHornerFMA}\,(p,x) - p(x)| \le \mathbf{u}|p(x)| + \gamma_n\gamma_{n+1}\,\widetilde{p}(x).$$

CompHornerFMA *requires $19n$ floating point operations.*

It is interesting to interpret the previous theorem with respect to the condition number of the polynomial evaluation of p at x. Combining the error bound in Theorem 4 with the condition number (3) for the polynomial evaluation gives the following relation,

$$\frac{|\mathsf{CompHornerFMA}\,(p,x) - p(x)|}{|p(x)|} \le \mathbf{u} + \gamma_n\gamma_{n+1}\,\mathrm{cond}(p,x). \qquad (9)$$

For practical purpose, just consider $\gamma_n\gamma_{n+1}$ as $\mathbf{u}^2$. In other words, the bound for the relative error of the computed result is essentially $\mathbf{u}^2$ times the condition number of the polynomial evaluation, plus the inevitable summand $\mathbf{u}$ for the final rounding of the result to the working precision. In particular, while $\mathrm{cond}(p,x) \lesssim 1/\mathbf{u}$, then the relative accuracy of the result is bounded by a constant of the order $\mathbf{u}$. This means that the compensated Horner algorithm computes an evaluation accurate to the last few bits as long as the condition number is smaller than $1/\mathbf{u}$. Besides that, Relation (9) tells us that the computed result is as accurate as if computed by the classic Horner algorithm with twice the working precision. Of course no accuracy can be expected for condition number larger than $1/\mathbf{u}^2$.

4.3. Compensating Horner

The principle of next algorithm CompHorner is the same as CompHornerFMA. Nevertheless we need an EFT which computes the rounding error generated by Horner, that is for polynomial evaluation without using the FMA. Next results provide this EFT.

Algorithm 8. EFT for Horner.

function $[q_0, p_\pi, p_\sigma] = \mathsf{EFTHorner}(p,x)$
$q_n = a_n$
for $i = n-1 : -1 : 0$
  $[p_i, \pi_i] = \mathsf{TwoProd}\,(q_{i+1}, x)$
  $[q_i, \sigma_i] = \mathsf{TwoSum}\,(p_i, a_i)$
  Let π_i be the coefficient of degree i in p_π
  Let σ_i be the coefficient of degree i in p_σ
end

Theorem 5. *Let $p(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial of degree n with floating point coefficients, and let x be a floating point value. Then following Algorithm 8 computes both*

- *the floating point value* $\mathsf{Horner}\,(p,x)$ *(Algorithm 4), and*
- *two polynomials p_π and p_σ, of degree $n-1$, with floating point coefficients;*

we write

$$[\mathsf{Horner}\,(p,x)\,, p_\pi, p_\sigma] = \mathsf{EFTHorner}\,(p,x)\,.$$

Algorithm 8 requires $8n$ flops.

We have the next EFT,

$$p(x) = \mathsf{Horner}\,(p,x) + (p_\pi + p_\sigma)(x), \qquad (10)$$

with

$$(\widetilde{p_\pi} + \widetilde{p_\sigma})(x) \le \gamma_{2n}\,\widetilde{p}(x).$$

We deduce another compensated evaluation algorithm based on the previous EFT for Horner.

Algorithm 9. Compensated Horner algorithm.

function $r = \mathsf{CompHorner}\,(p, x)$
$[h, p_\pi, p_\sigma] = \mathsf{EFTHorner}\,(p, x)$
$c = \mathsf{HornerFMA}\,(p_\pi \oplus p_\sigma, x)$
$r = h \oplus c$

Theorem 6. *Given a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ of degree n with floating point coefficients, and x a floating point value. We consider the result* CompHorner (p, x) *computed by Algorithm 9. Then,*

$$|\mathsf{CompHorner}\,(p, x) - p(x)| \leq \mathbf{u}|p(x)| + \gamma_n\gamma_{2n+1}\,\widetilde{p}(x).$$

Algorithm 9 requires $10n - 1$ floating point operations.

Again, combining the error bound in Theorem 6 with the condition number (3) for polynomial evaluation leads to

$$\frac{|\mathsf{CompHorner}\,(p, x) - p(x)|}{|p(x)|} \leq \mathbf{u} + \gamma_n\gamma_{2n+1}\,\mathrm{cond}(p, x). \tag{11}$$

Since $\gamma_n\gamma_{2n+1} \approx \mathbf{u}^2$, the previous remarks about error bound (9) also apply to the previous one. While CompHornerFMA needs almost two times more flop than CompHorner, we notice that the error bounds (9) and (11) are similar.

Actually, our next experimental results confirm that CompHorner is more efficient than CompHornerFMA in terms of computing time while being similarly accurate.

4.4. Experimental scheme

All our experiments are performed using IEEE-754 double precision. Since the double-doubles [8, 12] are usually considered as the most efficient portable library to double the IEEE-754 double precision, we consider it as a reference in the following comparisons. For our purpose, it suffices to know that a double-double number a is the pair (a_h, a_l) of IEEE-754 floating point numbers with $a = a_h + a_l$ and $|a_l| \leq \mathbf{u}|a_h|$. This property implies a renormalization step after each arithmetic operation. We denote by DDHorner our implementation of the Horner algorithm with the double-double format, derived from the implementation proposed by the authors of [12]. We notice that the double-double arithmetic naturally benefits from the availability of a FMA instruction: DDHorner uses TwoProd in the inner loop of the Horner algorithm. DDHorner requires $20n$ floating point operations. Using the double-double library proposed in [8], we can slightly reduce this flop count, but it has almost no impact on the measured computing times.

4.5. Accuracy tests

We test the expanded form of the polynomial $p_n(x) = (x-1)^n$. Accuracy of the evaluation is not guaranteed in the neighborhood of the real root 1 of p_n. Indeed the condition number is

$$\mathrm{cond}(p_n, x) = \frac{\widetilde{p_n}(x)}{|p_n(x)|} = \left|\frac{|x| + 1}{x - 1}\right|^n,$$

and $\mathrm{cond}(p_n, x)$ grows exponentially with respect to n. In the experiments reported on Figure 1, we have chosen $x = fl(1.333)$ to provide a binary floating point value with many non-zero bits in its mantissa. The value of $\mathrm{cond}(p_n, x)$ varies from 10^2 to 10^{40}, that corresponds to degrees n range 3 to 42. These huge condition numbers have a sense since the coefficients of p and the value x are floating point numbers.

We experiment both HornerFMA, CompHornerFMA, CompHorner and DDHorner (see Table 1). For every polynomial p_n, the exact value $p_n(x)$ is approximated with high accuracy thanks to the MPFR library [14]. Figure 1 presents the relative accuracy $|r - p_n(x)|/|p_n(x)|$ of the evaluation r computed by each algorithm. We set to the value one relative errors greater than one, which means that almost no useful information is available in the computed result. We also display the *a priori* error estimates (6) and (9) – no difference between (9) and (11) appears on this figure. We observe that our compensated algorithms exhibit the expected behavior: compensated results are roughly as if the Horner algorithm is computed with twice more bits. We identify no significant difference between the accuracy provided by double-double implementation compared to compensated ones. The full precision solution is computed as long as the condition number is smaller than $\mathbf{u}^{-1} \approx 10^{16}$. Then, for condition numbers between $\mathbf{u}^{-1}$ and $\mathbf{u}^{-2} \approx 10^{32}$, the relative error degrades (linearly in the log scale) to no accuracy at all as it was expected from the *a priori* error bounds (9) and (11).

As usual, these *a priori* bounds are definitely pessimistic especially when the condition number term becomes predominant in (9) and (11). More realistic bounds are provided by a dynamic analysis we describe in [7]. This latter reference also presents experiments with a more generic choice of polynomials; results are still similar to those displayed on Figure 1.

4.6. Running-time tests

All the algorithms are implemented in a C-code. We use the same programming techniques for the implementations of the three routines CompHornerFMA, CompHorner and DDHornerFMA. The experimental environments are listed in Table 3. Our measures are performed with polynomials whose degrees vary from 5 to 200 by step of 5. We randomly choose the values of the coefficients and the arguments. For each degree, the routines are tested on the same polynomial with the same argument. Table 4 displays the time overhead of the algorithms with respect to

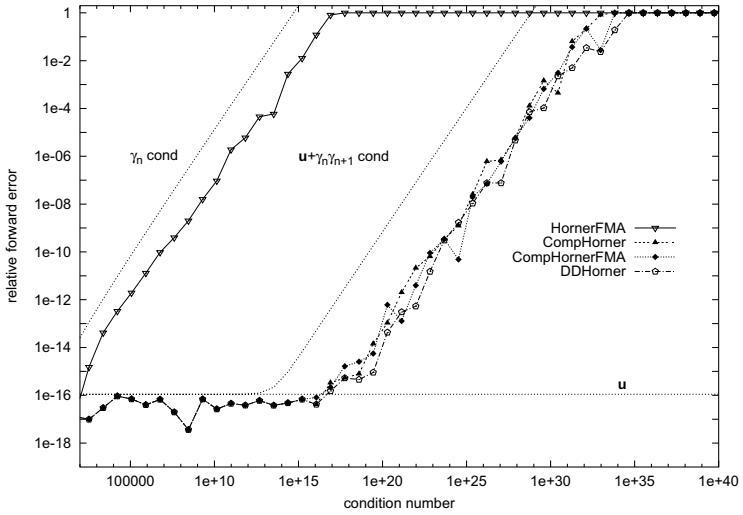


Figure 1. Accuracy of the polynomial evaluations.

HornerFMA. We have reported the minimum, the mean and the maximum of these ratios. The theoretical overheads (resulting from the number of floating point operations involved by each algorithm) are also reported.

Our compensated algorithms CompHornerFMA and CompHorner are both significantly faster than DDHorner. Algorithm CompHorner seems to be the most efficient alternative to improve the accuracy of the Horner algorithm. It runs about 1.8 times faster than CompHornerFMA and more than two times faster than DDHorner that uses the double-double library. We also notice that the measured overheads are always significantly smaller than theoretically expected. This issue will be explained in last Section 5.5.

5. Dot product

The purpose is now to compare the classic dot product algorithm without and with the use of FMA and corresponding compensated algorithms.

5.1. Classic dot product algorithm

Let $x = (x_1, \ldots, x_n)^T$ and $y = (x_1, \ldots, x_n)^T$ be n-vectors with floating point elements. The classic algorithm to compute a dot product is the following.

Algorithm 10. Dot product

function $s_n = \mathsf{Dot}\,(x, y)$
  $s_1 = x_1 y_1$
  for $i = 2 : n$
    $s_i = x_i y_i + s_{i-1}$
  end

Algorithm 10 requires $2n-1$ flops. The computed result satisfies [9]

$$|\mathsf{Dot}(x, y) - x^T y| \leq \gamma_n |x^T||y|. \tag{12}$$

FMA is suitable for dot product algorithm. We now look at the dot product algorithm where we use the FMA instead of the classic multiplication and addition.

Algorithm 11. Dot product with FMA.

function $s_n = \mathsf{DotFMA}\,(x, y)$
  $s_1 = x_1 y_1$
  for $i = 2 : n$
    $s_i = \mathsf{FMA}\,(x_i, y_i, s_{i-1})$
  end

As we can see, the number of floating point operations is divided by two when the FMA is used: algorithm DotFMA only requires n floating point operations. Nevertheless, the FMA does not improve the worst case accuracy of the computed dot product. Indeed, the two previous algorithms share the same error bound (12). Let us remark that the γ_n factor does not describe the number of floating point operations but the length of the largest path from the data to the result in the data flow graph.

5.2. Compensated DotFMA

Again $x = (x_1, \ldots, x_n)^T$ and $y = (x_1, \ldots, x_n)^T$ are two n-vectors with floating point elements. We consider the following compensated version of DotFMA. The rounding errors generated by every FMA are computed thanks to ThreeFMA (Algorithm 3).

Algorithm 12. Compensated DotFMA.

Table 3. Experimental environments

environment	description
I	Intel Itanium I, 733MHz, GNU Compiler Collection 2.96
II	Intel Itanium II, 1.5GHz, GNU Compiler Collection 3.4.6
III	Intel Itanium I, 733 MHz (16KB L1, 96KB L2 cache), Intel C++ Compiler v9.0.
IV	Intel Itanium II, 1.6 GHz (32KB L1, 256KB L2 cache), Intel C++ Compiler v9.0.

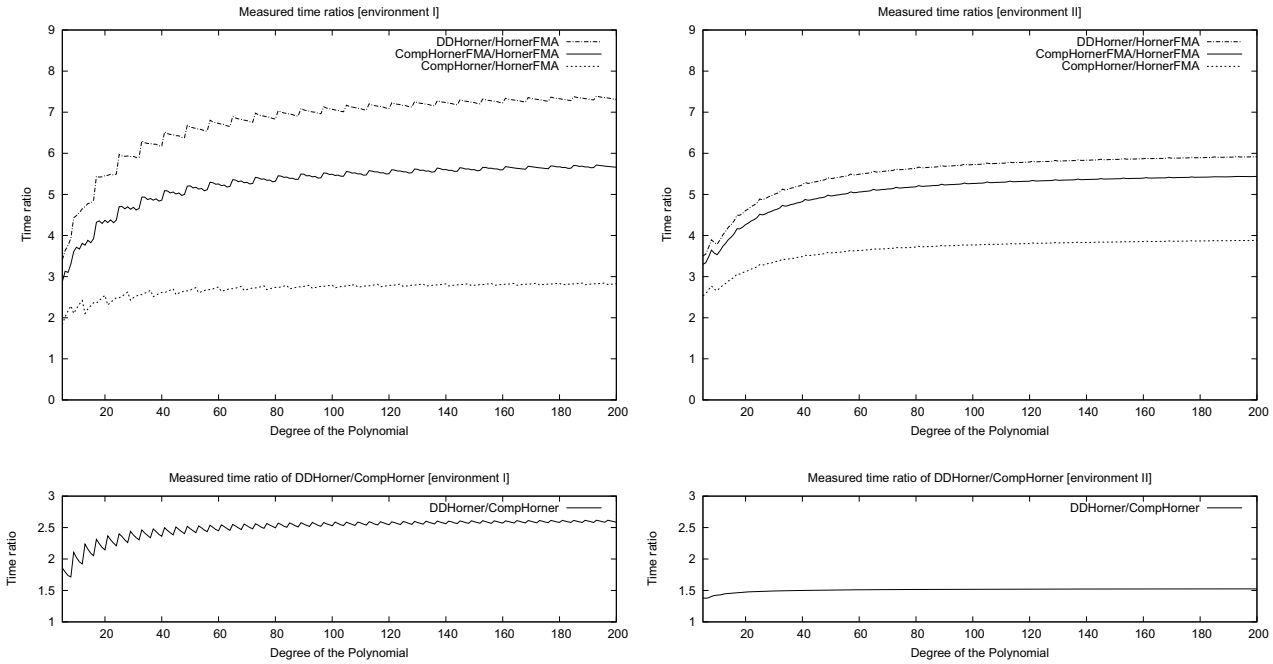


Figure 2. Measured overhead for CompHornerFMA**,** CompHorner **and** DDHorner **with respect to the polynomial degree (environment I on the left, and II on the right).**

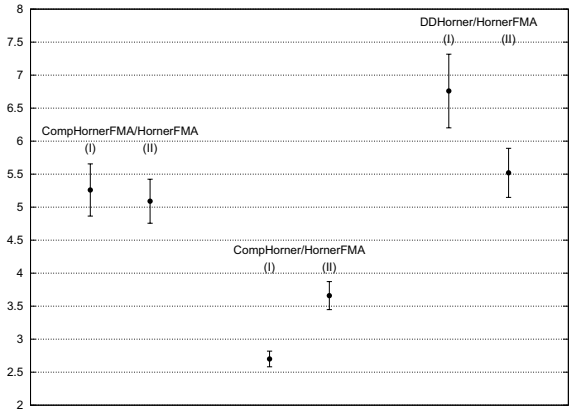


Figure 3. Overhead of CompHornerFMA**,** CompHorner **and** DDHorner **compared to** HornerFMA**: mean values are reported together with mean absolute deviation (obtained from Figure 2).**

Table 4. Measured running-time overhead compared to theoretical values for polynomial evaluation.

environment	CompHornerFMA/HornerFMA				CompHorner/HornerFMA				DDHorner/HornerFMA			
	min.	mean	max.	theo.	min.	mean	max.	theo.	min.	mean	max.	theo.
I	2.9	5.3	5.7	19	1.8	2.7	2.8	10	3.4	6.8	7.4	20
II	3.3	5.1	5.4	19	2.5	3.7	3.9	10	3.5	5.5	5.9	20

function $r = \mathsf{CompDotFMA}\,(x, y)$
  $[s_1, c_1] = \mathsf{TwoProd}\,(x_1, y_1)$
  for $i = 2 : n$
    $[s_i, \alpha_i, \beta_i] = \mathsf{ThreeFMA}\,(x_i, y_i, s_{i-1})$
    $c_i = c_{i-1} \oplus (\alpha_i \oplus \beta_i)$
  end
  $r = s_n \oplus c_n$

Proposition 7. *The result computed by previous Algorithm 12 satisfies*

$$|\mathsf{CompDotFMA}\,(x, y) - x^T y| \leq \mathbf{u}|x^T y| + \mathbf{u}\gamma_{n+1}|x|^T|y|. \quad (13)$$

CompDotFMA *requires* $19n-16$ *floating point operations.*

Proofs are detailed in [6].

5.3. Compensating Dot

The following algorithm for dot product computation is due to Ogita, Rump and Oishi [16].

Algorithm 13. Compensated Dot.

function $r = \mathsf{CompDot}\,(x, y)$
  $[s_1, c_1] = \mathsf{TwoProd}\,(x_1, y_1)$
  for $i = 2 : n$
    $[p_i, \pi_i] = \mathsf{TwoProd}\,(x_i, y_i)$
    $[s_i, \sigma_i] = \mathsf{TwoSum}\,(p_i, s_{i-1})$
    $c_i = c_{i-1} \oplus (\pi_i \oplus \sigma_i)$
  end
  $r = s_n \oplus c_n$

The following proposition sums up the properties of this algorithm.

Proposition 8 ([16]). *If no underflow occurs, the result computed by Algorithm 13 satisfies*

$$|\mathsf{CompDot}\,(x, y) - x^T y| \leq \mathbf{u}|x^T y| + \gamma_n^2|x^T||y|, \quad (14)$$

CompDot *algorithm requires* $10n - 7$ *flops when the* FMA *is available.*

Let us note again that this proposition means that the compensated algorithm returns a computed dot product as accurate as if computed in twice the working precision.

Experimental scheme for the following testing of accuracy and running-time issues is similar to the one we have described in Section 4.4.

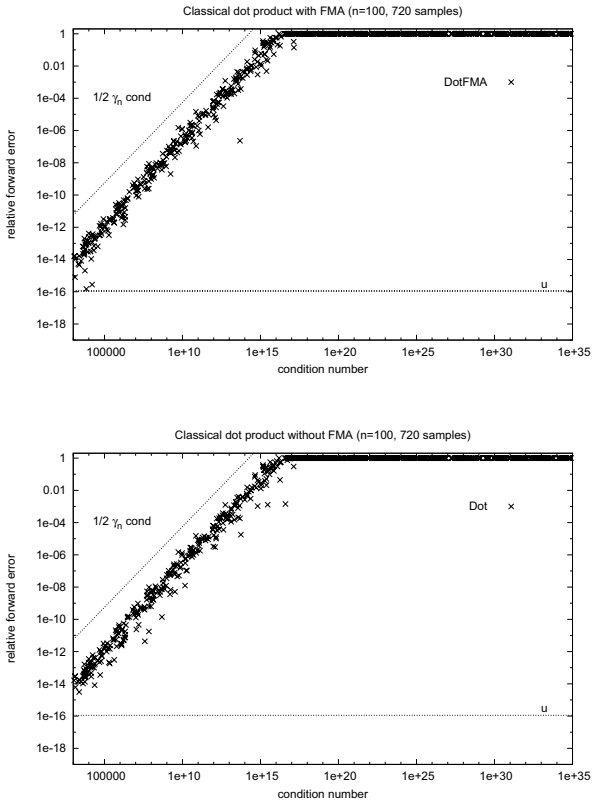


Figure 4. Accuracy of classic dot product algorithm with and without FMA

5.4. Accuracy Tests

For testing the actual accuracy reached by the various dot product algorithms previously presented, we need to generate dot products with condition number up to about 10^{32}. For this purpose, we use the random generator of ill-conditioned dot product GenDot described in [16]. Here it allows us to generate 720 dot products of length $n = 100$, with condition numbers varying from 10^2 to 10^{35}. We always use the same set of dot products in all our experiments.

Figure 4 presents the results for classic dot product algorithm with and without FMA. We display the relative error of the computed result with respect to the condition num-

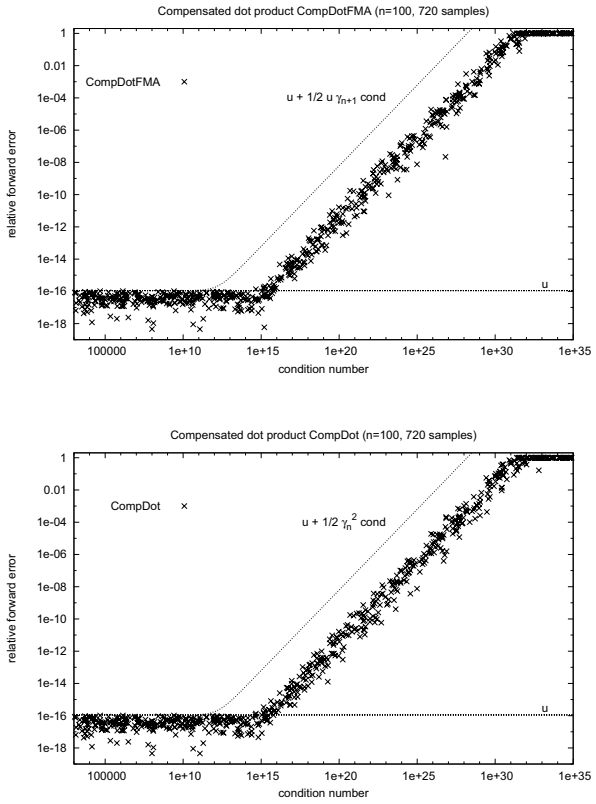


Figure 5. Accuracy of compensated dot products CompDotFMA **and** CompDot

ber. The dashed curves represent the relative error bounds derived from Relations (13) and (14). As we can see, the use of FMA does not significantly improve the accuracy of the result. Even if the theoretical bounds are pessimistic, they provide a reasonable estimate of the actual error bounds.

Figure 5 presents the results for compensated dot products CompDotFMA and CompDot. Again this figure illustrates that using the error-free transformation ThreeFMA does not improve the accuracy of the result compared to the error-free transformation TwoProd. Finally the accuracy of the considered algorithms does not benefit from the use of FMA as it was expected from the theoretical worst case bounds.

5.5. Running-time tests

The measured execution times are reported with Table 5. The timings are compared with ordinary dot product algorithm DotFMA. Last row also reports the theoretical ratios.

These results show that the compensated algorithms CompDotFMA and CompDot run both considerably faster than DDDot.

Table 5. Running-time ratios of dot products. Environment III (top) and IV (middle) are compared to theoretical flop counts (bottom) for various vector lengths n.

n	CompDot / DotFMA	CompDotFMA / DotFMA	DDDot / DotFMA
50	1.4	2.3	8.24
100	1.29	2.37	8.98
1000	1.24	2.63	10.46
10000	1.25	2.63	10.5
100000	1.07	1.76	6.27
50	1.63	2.61	9.87
100	1.35	2.43	9.65
1000	1.26	2.6	10.86
10000	1.25	2.62	10.97
100000	1.25	2.35	9.8
Theoret.	10	19	22

As previously observed for polynomial evaluation, the measured ratios of the compensating process overhead are always smaller than the theoretical values. Theoretical ratios just count the floating point operations and do not take into account the complex instruction reordering the compiler or the processor perform. Most modern processors are capable of executing several instructions in parallel, but it is not always easy to exploit this feature in real programs. In order to exploit the ability to perform multiple instructions in parallel, both the compiler and the processor must reconstruct the implicit parallelism in a program which is usually written in a serial fashion. In particular, the main part of the instruction scheduling is performed by the compiler to take advantage of the instruction-level parallelism on Intel Itanium architecture. On the other hand, the possibility of performing parallel execution of instructions is not only limited by the architecture and the compiler performances, but also by the instruction-level parallelism which is an intrinsic parameter of the algorithm. For instance a program may require long sequences of serial instructions that can not be performed in parallel with any other. Compensated algorithms here exhibit a better intrinsic instruction-level parallelism than double-double ones since they are implemented with no normalization step.

6. Acknowledgment

Authors thank T. Ogita, S.M. Rump and S. Oishi for [16] that motivates the analysis and development of compensated algorithms. They also thank S. Graillat (LIP6, UPMC Paris) for his contribution to previous results about compensated algorithms.

References

[1] High-precision software directory. URL = `http://crd.lbl.gov/~dhbailey/mpdist`.

[2] D. H. Bailey. Algorithm 719: Multiprecision translation and execution of Fortran programs. *ACM Trans. Math. Software*, 19(3):288–319, 1993.

[3] S. Boldo and J.-M. Muller. Some functions computable with a fused mac. In IEEE, editor, *IEEE Symposium on Computer Arithmetic ARITH'17*, Cape Cod, Massachusetts, USA, June 2005.

[4] R. P. Brent. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Softw.*, 4(1):57–70, 1978.

[5] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18:224–242, 1971.

[6] S. Graillat, P. Langlois, and N. Louvet. Accurate dot products with FMA. In G. Hanrot and P. Zimmermann, editors, *RNC-7, Real Numbers and Computer Conference, Nancy, France*, pages 141–142, July 2006. Extended version available on-line.

[7] S. Graillat, P. Langlois, and N. Louvet. Fused Multiply and Add implementations of the compensated Horner scheme. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, Dagstuhl Seminar 6021, Jan. 2006. Extended version available on-line.

[8] Y. Hida, X. S. Li, and D. H. Bailey. Quad-double arithmetic: Algorithms, implementation, and application. In N. Burgess and L. Ciminiera, editors, *15th IEEE Symposium on Computer Arithmetic*, pages 155–162. Institute of Electrical and Electronics Engineers, June 2001.

[9] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[10] IEEE Computer Society, New York. *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985*, 1985. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

[11] D. E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. Addison-Wesley, Reading, MA, USA, third edition, 1998.

[12] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, B. J. Thompson, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. *ACM Transactions on Mathematical Software*, 28(2):152–205, June 2002.

[13] P. Markstein. *IA-64 and elementary functions. Speed and precision*. Hewlett-Packard Professionnal Books. Prentice-Hall PTR, 2000.

[14] The MPFR library. URL = `http://www.mpfr.org/`.

[15] Y. Nievergelt. Scalar fused multiply-add instuctions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Transactions on Mathematical Software*, 29(1), Mar. 2003.

[16] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26(6):1955–1988, 2005.

Validated computation for infinite dimensional eigenvalue problems

Kaori Nagatou
Faculty of Mathematics
Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
nagatou@math.kyushu-u.ac.jp

Abstract

In this paper we will show how guaranteed bounds for eigenvalues (together with eigenvectors) are obtained and how non-existence of eigenvalues in a concrete region could be assured. Some examples for several types of operators in bounded and unbounded domains will be presented. We will furthermore discuss possible future applications to eigenvalue enclosing/excluing of Schrödinger operator, hopefully in its spectral gaps.

1 Introduction

Up to now we have developed a method to enclose and exclude eigenvalues for differential operators [10, 11, 12, 13, 14, 15]. This method is based on Nakao's theory known as a numerical verification method for partial differential equations [16, 17, 18, 19], and it has a merit that it could be applied even in case the operator is not self-adjoint. A remarkable point of this eigenvalue enclosing/excluding is to assure an existence and non-existence range of eigenvalues with mathematically rigorous sense. This means not only a reliability of computed eigenpairs but also that such evaluation of eigenvalues (and eigenvectors) can be applied to related another problems, e.g. another numerical verification methods for nonlinear problems or stability analysis of bifurcation phenomenon in hydrodynamics.

This paper aims to show how eigenvalues (and eigenvectors) are enclosed or excluded in mathematically rigorous sense. At first in Section 2, we briefly overview what the spectrum is, and in Section 3 we introduce some eigenvalue enclosure methods, especially for symmetric operators. Our original method is described in Section 4 together with some applications to algebla, another numerical verification methods for nonlinear problems and stability analysis of bifurcation phenomenon in hydrodinamics. Finally in Section 5, we will briefly comment our recent challenge to enclose or exclude eigenvalues in *spectral gaps* of one dimensional Schrödinger operator.

2 Spectrum and eigenvalue

Let X be a Banach space. We denote a set of linear and continuous operators on X by $\mathcal{L}(X)$. For simplicity, we consider a (linear) closed operator T on X and let $\mathcal{D}(T)$ denote a domain of definition.

We call $\rho(T) \equiv \{z \in \mathbf{C} \mid (T - zI)^{-1} \in \mathcal{L}(X)\}$ as a **resolvent set** of T, here I is an identity operator on X, and its complement in $\mathbf{C}$: $\sigma(T) = \mathbf{C} \backslash \rho(T)$ is called as **spectrum** of T. The spectrum could be further divided into three subsets as follows:

$$\begin{aligned}
\sigma_P(T) &= \{z \in \sigma(T) \mid T - zI \text{ has no inverse}\} \\
\sigma_C(T) &= \{z \in \sigma(T) \mid (T - zI)^{-1} \text{ has an unbounded} \\
&\qquad \text{inverse with domain dense in } X\} \\
\sigma_R(T) &= \{z \in \sigma(T) \mid (T - zI)^{-1} \text{ has an inverse} \\
&\qquad \text{(bounded or not) whose domain is not} \\
&\qquad \text{dense in } X\}
\end{aligned}$$

We call $\sigma_P(T)$, $\sigma_C(T)$ and $\sigma_R(T)$ as **point spectrum**, **continuous spectrum** and **residual spectrum**, respectively. The union of $\sigma_C(T)$ and $\sigma_R(T)$ could be called as **essential spectrum**. The necessary and sufficient condition for $\lambda \in \sigma_P(T)$ is that there exists $u \in \mathcal{D}(T)$ which is not identically zero and satisfies the relation $Tu = \lambda u$. We call $\lambda \in \sigma_P(T)$ as an **eigenvalue** of T and corresponding function $0 \neq u \in \mathcal{D}(T)$ is called as an **eigenvector**.

In case that X is finite dimensional, any linear operator A on X is represented by a matrix. Any point in $\sigma_P(A)$ corresponds to an eigenvalue of the matrix A and

$$\sigma_C(A) = \sigma_R(A) = \phi$$

holds. Therefore the spectrum is equal to eigenvalue in finite dimensional case, but infinite dimensional case (i.e. $\dim X = \infty$) there could be a big *gap* between spectrum itself and eigenvalue.

Example 1: Let X be a set of continous functions on an open interval (a, b), and consider two domains as follows:

$$\begin{aligned} D_1 &\equiv \{x \in X \mid x' \in X\}, \\ D_2 &\equiv \{x \in D_1 \mid x(a) = 0\}. \end{aligned}$$

Then two operators

$$T_1 : x \in D_1 \mapsto x'$$

and

$$T_2 : x \in D_2 \mapsto x'$$

which only differ concerning domain of definition, have the following different structure of spectrum:

$$\sigma(T_1) = \sigma_P(T_1) = \mathbf{C}, \quad \sigma(T_2) = \phi.$$

Example 2: Let X be a set of continuous functions on an interval $[0, 1]$ and define an operator on X

$$(Au)(x) \equiv \int_0^x u(y)dy \ (0 \le x \le 1).$$

Then we have

$$\sigma(A) = \sigma_C(A) = \{0\}.$$

As we can see in Example 1, the spectrum of a differential operator $x \mapsto x'$ depends on boundary conditions. The operator A in Example 2 is compact and it is known that a compact operator has rather similar nature of an operator in *finite* dimensional space. But this operator A has no point spectrum (i.e. eigenvalue) and it is one of the typical phenomenun in *infinite* dimensional case.

There are many interesting topics on spectral theory. See [4, 8] for example.

3 Enclosure methods for symmetric operators

Finding an eigenvalue and eigenvector of infinite dimensional operator (i.e. an operator defined in an infinite dimensional space) is called as *infinite dimensional eigenvalue problem*.

Let H be an infinite dimensional Hilbert space with an inner product $< \cdot, \cdot >$. For a linear symmetric operator $L : \mathcal{D}(L) \to H$, we consider the eigenvalue problem

$$Lu = \lambda u, \ \ u \in \mathcal{D}(L) \backslash \{0\}. \tag{3.1}$$

An eigenvalue of an operator takes an important role to understand a nonlinear phenomenon in science and engineering. Especially, it often becomes a key value when we consider a behavior of dynamical systems.

Several methods to enclose eigenvalues for symmetric operators have been proposed. In the below we introduce some of those methods. Here we assume that all eigenvalues are bounded below and ordered as $\lambda_1 \le \lambda_2 \le \cdots$.

Krylov-Weinstein's bounds [4]

As one of the simplest way of eigenvalue enclosure, Krylov-Weinstein's bounds is well known.

Let $(\widetilde{u}, \widetilde{\lambda}) \in \mathcal{D}(L) \times \mathbf{R}$ be an approximate eigenpair and compute

$$\delta \equiv \frac{\|L\widetilde{u} - \widetilde{\lambda}\widetilde{u}\|}{\|\widetilde{u}\|}.$$

Then the interval

$$[\widetilde{\lambda} - \delta, \widetilde{\lambda} + \delta]$$

contains at least one eigenvalue of L.

This bound is easy to compute, but the width of the enclosed interval is not so narrow. And it also has another defect that no information is obtained concerning the index of eigenvalue.

Kato-Temple's bounds [8]

As an improved version of Krylov-Weinstein's bounds, there is a Kato-Temple's bounds which was proposed in 1949.

Let $(\widetilde{u}, \widetilde{\lambda})$ be an approximate eigenpair satisfying

$$\widetilde{\lambda} = < L\widetilde{u}, \widetilde{u} > / < \widetilde{u}, \widetilde{u} >$$

and compute

$$\delta \equiv \frac{\|L\widetilde{u} - \widetilde{\lambda}\widetilde{u}\|}{\|\widetilde{u}\|}.$$

For the nth eigenvalue λ_n with finite multiplicity, suppose that an open interval (α, β) does not contain any spectrum except for λ_n. Then for $\rho \in \mathbf{R}$ satisfying $\alpha < \rho < \beta$, we have

$$\lambda_n \in \left[\rho - \frac{\delta^2}{\beta - \rho}, \rho + \frac{\delta^2}{\rho - \alpha}\right]. \tag{3.2}$$

The quality of this bounds is better than Krylov-Weinstein's bounds. Indeed it has an $O(\delta^2)$ quality compared with an $O(\delta)$ quality of Krylov-Weinstein's bounds. But it also has a difficulty that it needs a precise information on eigenvalue distribution in advance, i.e. a (rough) upper bound for λ_{n-1} and (rough) lower bound for λ_{n+1} are needed to obtain the result.

Rayleigh-Ritz bounds [4]

The Rayleigh-Ritz method is well known as a method to obtain very accurate upper bounds for the first N eigenvalues of L.

Let $\widetilde{u}_1, ..., \widetilde{u}_N \in \mathcal{D}(L)$ be linearly independent functions and define two $N \times N-$ matrices

$$\begin{aligned} A_1 &\equiv (< L\widetilde{u}_i, \widetilde{u}_j >)_{i,j=1,...,N}, \\ A_2 &\equiv (< \widetilde{u}_i, \widetilde{u}_j >)_{i,j=1,...,N}. \end{aligned}$$

Then, for N eigenvalues Λ_i $(i = 1, ..., N)$ of the matrix eigenvalue problem

$$A_1 x = \Lambda A_2 x,\ x \in \mathbf{R}^N \backslash \{0\}, \tag{3.3}$$

we have

$$\lambda_i \leq \Lambda_i \ (i = 1, ..., N). \tag{3.4}$$

Being different from Kato-Temple's bounds, this method does not need any a priori information concerning eigenvalue distribution, although it does not give any lower bounds.

Lehman's Bounds [3]

Concerning the lower bounds for eigenvalues, there is a Leman's method as follows.

Let $\widetilde{u}_1, ..., \widetilde{u}_N \in \mathcal{D}(L)$ be linearly independent functions and suppose that $\Lambda_N < \nu \leq \lambda_{N+1}$ holds for a real number ν, where Λ_N denotes the Rayleigh-Ritz bound. Moreover define three $N \times N-$ matrices

$$\begin{aligned} A_3 &\equiv (< L\widetilde{u}_i, L\widetilde{u}_j >)_{i,j=1,...,N}, \\ B_1 &\equiv A_1 - \nu A_2, \\ B_2 &\equiv A_3 - 2\nu A_1 + \nu^2 A_2, \end{aligned}$$

where A_1 and A_2 are the same matrices in Rayleigh-Ritz method. Then, for N eigenvalues μ_i $(i = 1, ..., N)$ of the matrix eigenvalue problem

$$B_1 x = \mu B_2 x,\ x \in \mathbf{R}^N \backslash \{0\}, \tag{3.5}$$

we have

$$\lambda_{N+1-i} \geq \nu + \frac{1}{\mu_i} \ (i = 1, ..., N). \tag{3.6}$$

This lower bound is also sharp, but it also has the same difficulty as Kato-Temple's method, i.e. it needs a priori information on the exact eigenvalue λ_{N+1}.

Homotopy Method [20]

In order to overcome the difficulty to obtain a priori information on exact eigenvalues, the homotopy method was proposed by Plum in 1990. In his method a base problem is considered which corresponds to the given problem, i.e. for the eigenvalue problem for L. Here the base problem is chosen so that the eigenvalue distribution of it is already obtained. Let L_0 be an operator which corresponds to this base problem, then consider a homotopy which connects two operators L and L_0:

$$L_s \equiv (1 - s)L_0 + sL, \ \ s \in [0, 1].$$

Then starting from $s = 0$ and making use of the continuity and monotonicity of eigenvalues on the parameter s, some eigenvalues for L_s are enclosed in each step. Finally the first several eigenvalues of L are enclosed when the parameter s reached 1.

Besides these methods, there is an intermediate methods [1, 2], but all these methods are restricted to symmetric operators and cannot be applied to non-symmetric operators. Moreover, any eigenvectors are not enclosed by these methods. In the next section, we introduce our method which could be also applied to non-symmetric operators and also provides the eigenvector enclosures.

4 Enclosure method based on Nakao's theory

We have developed a method to enclose eigenvalues and eigenvectors for differential operators [10, 11, 12], which was based on Nakao's verification methods for nonlinear differential equations [16, 17, 18, 19]. Our method is also applicable to non-symmetric operators. So far we have applied our enclosure method to enclose eigenpair of symmetric operators and to enclose real eigenvalues and corresponding eigenvectors of a non-symmetric operator.

Now we describe the principle of our method, and show some applications.

4.1 Eigenvalue enclosing and excluding method ([10, 11, 12])

Though the following arguments are almost the same as [12], in order to keep this paper to be self-contained, we will give the detailed description in the below.

We consider a self-adjoint eigenvalue problem:

$$\begin{cases} -\Delta u + qu &= \lambda u \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases} \tag{4.1}$$

Here Ω is a bounded convex domain in $\mathbf{R}^2$ and let $q \in L^\infty(\Omega)$. We apply Nakao's method which is known as a numerical verification method for nonlinear problems.

In what follows, for some integer m, let $H^m(\Omega)$ denote the L^2-Sobolev space of order m on Ω. Then, define $H_0^1(\Omega) \equiv \{v \in H^1(\Omega) \mid v = 0 \text{ on } \partial\Omega\}$ with the inner product $< u, v >_{H_0^1} \equiv (\nabla u, \nabla v)_{L^2}$ for $u, v \in H_0^1(\Omega)$, and the norm $\|u\|_{H_0^1} \equiv \|\nabla u\|_{L^2}$ for $u \in H_0^1(\Omega)$, where $(\cdot, \cdot)_{L^2}$ and $\|\cdot\|_{L^2}$ represent the inner product and the norm on $L^2(\Omega)$, respectively.

Now, let S_h be a finite dimensional subspace of $H_0^1(\Omega)$ dependent on h $(0 < h < 1)$. Usually, S_h is taken to be a finite element subspace with mesh size h. Also, let

$$P_{h0} : H_0^1(\Omega) \longrightarrow S_h$$

denote the H_0^1-projection defined by

$$(\nabla(u - P_{h0}u), \nabla v)_{L^2} = 0 \qquad \text{for all } v \in S_h.$$

We now assume the following approximation property in S_h :

Assumption 1. For any $u \in H^2(\Omega) \bigcap H_0^1(\Omega)$,

$$\inf_{\chi \in S_h} \|u - \chi\|_{H_0^1} \leq C_1 h |u|_{H^2}, \tag{4.2}$$

where

$$|u|_{H^2}^2 \equiv \sum_{i,j=1}^{2} \|\frac{\partial^2 u}{\partial x_i \partial x_j}\|_{L^2}^2.$$

Here, C_1 is a positive, numerically determined constant which is independent of h.

The following lemma is well known [6]:

Lemma 1. *For any ψ in $L^2(\Omega)$, there exists a unique solution $\phi \in H^2(\Omega) \bigcap H_0^1(\Omega)$ of the following Poisson equation:*

$$\begin{cases} -\Delta\phi &= \psi \quad in\ \Omega, \\ \phi &= 0 \quad on\ \partial\Omega. \end{cases} \tag{4.3}$$

Furthermore, there exists a positive constant C_2 satisfying

$$|\phi|_{H^2} \leq C_2 \|\psi\|_{L^2}. \tag{4.4}$$

In particular, if Ω is a convex polygonal domain, we can set $C_2 = 1$ ([6]).

Since we want to verify the eigenpairs of this problem, we consider the space $H_0^1(\Omega) \times R$, and define the innner product $< \cdot, \cdot >_{H_0^1 \times R}$ and the norm $\| \cdot \|_{H_0^1 \times R}$ by

$$< w_1, w_2 >_{H_0^1 \times R} \equiv (\nabla u_1, \nabla u_2)_{L^2} + \lambda_1 \lambda_2,$$

$$\|w\|_{H_0^1 \times R} \equiv (\|u\|_{H_0^1}^2 + |\lambda|^2)^{\frac{1}{2}},$$

respectively, where $w_i = (u_i, \lambda_i) \in H_0^1(\Omega) \times R$ $(i = 1, 2)$ and $w = (u, \lambda) \in H_0^1(\Omega) \times R$. Moreover, let I_0 and I be the identity map on $H_0^1(\Omega)$ and $H_0^1(\Omega) \times R$, respectively.

We first normalize the problem (4.1) as

$$\text{find } (\widehat{u}, \lambda) \in H_0^1(\Omega) \times R \text{ s.t.}$$

$$\begin{cases} -\Delta\widehat{u} + (q - \lambda)\widehat{u} &= 0, \\ \displaystyle\int_\Omega \widehat{u}^2\, dx &= 1. \end{cases} \tag{4.5}$$

We define the projection

$$P_h : H_0^1(\Omega) \times R \longrightarrow S_h \times R$$

by

$$P_h(u, \lambda) \equiv (P_{h0}u, \lambda).$$

Now, let $\widehat{w}_h = (\widehat{u}_h, \widehat{\lambda}_h) \in S_h \times R$ be a finite element solution of (4.5), that is,

$$\begin{cases} (\nabla\widehat{u}_h, \nabla\phi_i)_{L^2} &= ((\widehat{\lambda}_h - q)\widehat{u}_h, \phi_i)_{L^2} \qquad \forall \phi_i \in S_h, \\ \displaystyle\int_\Omega \widehat{u}_h^2\, dx &= 1. \end{cases} \tag{4.6}$$

We will verify the existence of the eigenvalues and the eigenfunctions for (4.5) in the neighborhood of $(\bar{u}, \widehat{\lambda}_h)$ satisfying

$$\begin{cases} -\Delta\bar{u} + (q - \widehat{\lambda}_h)\widehat{u}_h &= 0 \quad \text{in } \Omega, \\ \bar{u} &= 0 \quad \text{on } \partial\Omega. \end{cases} \tag{4.7}$$

Notice that $\bar{u} \in H^2(\Omega) \bigcap H_0^1(\Omega)$, and $\widehat{w}_h = P_h(\bar{u}, \widehat{\lambda}_h)$. We have by (4.5) and (4.7)

$$\begin{cases} -\Delta(\widehat{u} - \bar{u}) &= (\lambda - q)\widehat{u} - (\widehat{\lambda}_h - q)\widehat{u}_h, \\ \displaystyle\int_\Omega \widehat{u}^2\, dx &= 1. \end{cases} \tag{4.8}$$

Defining $v_0 = \bar{u} - \widehat{u}_h$, we then have $v_0 \in S_h^\perp$, where $S_h^\perp$ means the orthogonal complement of S_h in $H_0^1(\Omega)$, and we can write

$$\bar{u} = \widehat{u}_h + v_0 \quad \text{ for } \widehat{u}_h \in S_h \text{ and } v_0 \in S_h^\perp.$$

Here we use a posteriori estimates for v_0 as below.

Let $S_h^* \subset H^1(\Omega)$ be a finite element subspace whose basis consists of the union of the basis on S_h and the base functions having nonzero values on the boundary $\partial\Omega$. Define $\bar{\nabla}\widehat{u}_h \in S_h^* \times S_h^*$, a vector function in two dimension, by the L^2-projection of $\nabla\widehat{u}_h \in L^2 \times L^2$ to $S_h^* \times S_h^*$. Then, define $\bar{\Delta}\widehat{u}_h \in L^2(\Omega)$ by

$$\bar{\Delta}\widehat{u}_h \equiv \nabla \cdot \bar{\nabla}\widehat{u}_h.$$

We then obtain the following estimation (*cf.*[25]):

$$\|v_0\|_{H_0^1} \equiv \|\nabla\widehat{u}_h - \bar{\nabla}\widehat{u}_h\| + C_0 h \|\bar{\Delta}\widehat{u}_h + (\widehat{\lambda}_h - q)\widehat{u}_h\|,$$

where $C_0 \equiv C_1 C_2$. Using the well-known Aubin-Nitsche trick ([9]), we can estimate the L^2 norm of v_0 as

$$\|v_0\|_{L^2} \leq C_0 h \|v_0\|_{H_0^1}.$$

Now, in order to verify solutions $(\widehat{u}, \lambda)$ of (4.5) near $(\bar{u}, \widehat{\lambda}_h)$, representing

$$\widehat{u} = \bar{u} + \widetilde{u}\ ,\ \lambda = \widehat{\lambda}_h + \widetilde{\lambda},$$

we can rewrite (4.8) as

$$\begin{aligned} -\Delta\widetilde{u} &= (\widehat{\lambda}_h + \widetilde{\lambda} - q)(\widetilde{u} + \widehat{u}_h + v_0) \\ &\quad -(\widehat{\lambda}_h - q)\widehat{u}_h, \\ \int_\Omega (\widetilde{u} + \widehat{u}_h + v_0)^2\, dx &= 1. \end{aligned}$$

Thus using the following compact map on $H_0^1(\Omega) \times R$

$$\begin{aligned} F(\widetilde{u}, \widetilde{\lambda}) \equiv \big((-\Delta)^{-1}\{&(\widehat{\lambda}_h + \widetilde{\lambda} - q)(\widetilde{u} + \widehat{u}_h + v_0) \\ &- (\widehat{\lambda}_h - q)\widehat{u}_h\}, \\ &\widetilde{\lambda} + \int_\Omega (\widetilde{u} + \widehat{u}_h + v_0)^2\, dx - 1\big), \end{aligned} \tag{4.9}$$

where $(-\Delta)^{-1}$ means the solution operator for Poisson equation with homogeneous boundary condition, we have the fixed point equation for $w = (\widetilde{u}, \widetilde{\lambda})$

$$w = F(w). \tag{4.10}$$

We now assume the following assumption.

Assumption 2. Set $\rho \equiv (-v_0, 0)$ and define $F'(\rho)$ as the Fréchet derivative of F at ρ. Assume that restriction to $S_h \times R$ of the operator $P_h[I - F'(\rho)] : H_0^1(\Omega) \times R \longrightarrow S_h \times R$ has an inverse

$$[I - F'(\rho)]_h^{-1} : S_h \times R \longrightarrow S_h \times R.$$

This assumption can be numerically checked in the actual computation.

Now we decompose (4.10) into the finite and the infinite dimensional parts:

$$\begin{cases} P_h w &= P_h F(w), \\ (I - P_h)w &= (I - P_h)F(w). \end{cases} \tag{4.11}$$

And we use the Newton-like method only for the former part of (4.11), that is, we define the Newton-like operator

$$N_h(w) \equiv P_h w - [I - F'(\rho)]_h^{-1}(P_h w - P_h F(w)).$$

We next define the operator

$$T : H_0^1(\Omega) \times R \longrightarrow H_0^1(\Omega) \times R$$

as

$$T(w) \equiv N_h(w) + (I - P_h)F(w). \tag{4.12}$$

Then T becomes a compact map on $H_0^1(\Omega) \times R$, and

$$w = T(w) \Longleftrightarrow w = F(w) \tag{4.13}$$

holds.

An arbitrary element $w \in H_0^1(\Omega) \times R$ can be uniquely written as

$$w = (v_h, \mu) + (v_\perp, 0),\ (v_h, \mu) \in S_h \times R,\ (v_\perp, 0) \in S_h^\perp \times \{0\} \tag{4.14}$$

with

$$v_h = \sum_{j=1}^{M} v_j \phi_j,\ M = \dim S_h.$$

And for w in (4.14) we use the following notation:

$$\begin{aligned} (w)_i &\equiv |v_i|, i = 1, ..., M, \\ (w)_{M+1} &\equiv \|v_\perp\|_{H_0^1}, \\ (w)_{M+2} &\equiv |\mu| \end{aligned}$$

Now, we intend to find a solution to (4.5) in a set W, referred to as a 'candidate set'. Taking a vector $(W_1, ..., W_{M+2})^t$ such that $W_i > 0 (i = 1, ..., M + 2)$, a candidate set W is defined by

$$W \equiv \{w \in H_0^1(\Omega) \times R \mid (w)_i \le W_i\ (i = 1, ..., M + 2)\} \tag{4.15}$$

Now let T' be the Fréchet derivative of T. Then we choose two vectors

$$(Y_1, ..., Y_{M+2})^t, Y_i > 0 \quad (i = 1, ..., M + 2)$$

and

$$(Z_1, ..., Z_{M+2})^t, Z_i > 0 \quad (i = 1, ..., M + 2)$$

such that

$$\begin{aligned} (T(0))_i &\le Y_i,\ i = 1, ..., M + 2, \\ (T'(w_1)w_2)_i &\le Z_i,\ i = 1, ..., M + 2, \\ &\quad \forall w_1, w_2 \in W \end{aligned}$$

The verification condition is described in the following theorem.

Theorem 1. *If a candidate set W, defined by (4.15), satisfies*

$$Y_i + Z_i < W_i\ \ (i = 1, ..., M + 2), \tag{4.16}$$

then there exists a fixed point of T in

$$K \equiv \{v \in H_0^1(\Omega) \times R \mid (v)_i \le Y_i + Z_i\ (i = 1, ..., M + 2)\}. \tag{4.17}$$

Moreover, this fixed point is unique within the set W.

By this method we can uniquely enclose an eigenpair $(\widehat{u}, \lambda)$ in the set W. The author further extended this method by using an infinite dimensional homotopy method, and obtained the local uniqueness of eigenvalue and eigenvector *respectively* as follows:

Theorem 2. *If a set $W = U \times R$ satisfies the conditions in Theorem 1, then we have*

i) $^{\exists 1}u^*$ *:eigenfunction s.t.* $u^* - \bar{u} \in U$, $\displaystyle\int_\Omega (u^*)^2\,dx = 1$,

ii) $^{\exists 1}\lambda^*$ *:eigenvalue s.t.* $\lambda^* - \widehat{\lambda}_h \in \Lambda$,

iii) $F(u^* - \bar{u}, \lambda^* - \widehat{\lambda}_h) = (u^* - \bar{u}, \lambda^* - \widehat{\lambda}_h)$,

iv) λ^* *:geometric simple eigenvalue.*

(See [12] for the proof of Theorem 1 and Theorem 2.)

It is remarkable that this extended method can assure that the enclosed eigenvalue is simple in mathematically rigorous sense.

Moreover we have proposed a method to *exclude* an eigenvalue in a concrete interval, i.e. to prove that there is no eigenvalue in such an interval. This could be done as follows.

Let Λ be a narrow interval in which we want to exclude any eigenvalues. Then consider the linear equation

$$\begin{cases} -\Delta u + qu &= \Lambda u \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases} \tag{4.18}$$

Since the equation (4.18) has a trivial solution $u \equiv 0$, if we could prove the uniqueness of the solution of (4.18) then the non-existance of eigenvalues in Λ could be confirmed. We can also apply Nakao's method to enclose the unique solution of (4.18). (See [10, 11] for details.) These enclosing and excluding methods will be able to apply to the problem in $\mathbf{R^3}$.

4.2 Applications of our method

Now we present some examples to which we have applied our method so far.

Application 1: A numerical verification of solutions for nonlinear elliptic problems [11]

We consider the nonlinear elliptic boudary value problem:

$$\begin{cases} -\Delta u &= f(u) \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{cases} \tag{4.19}$$

where Ω is a bounded convex domain in $\mathbf{R}^2$ and $f : H_0^1(\Omega) \to L^2(\Omega)$ satisfies some suitable conditions. (cf. [11]). In [11] we evaluated the norm of the inverse operator of the linearized operator by making use of our eigenvalue excluding method, and used the infinite dimensional Newton's method which is based on Plum's method [21]. This can be regarded as a combined method between Nakao's method and Plum's method.

Application 2: Linearized eigenvalue problem at an exact solution of nonlinear problems [13]

We consider the follwing problem:

Find $(u, v, \lambda) \in H_0^1(\Omega) \times H_0^1(\Omega) \times \mathbf{R}$ s.t.

$$\begin{cases} -\Delta u &= f(u), \\ -\Delta v - f'(u)v &= \lambda v, \\ \displaystyle\int_\Omega v^2\,dx &= 1, \end{cases} \tag{4.20}$$

where Ω and f are same as in Application 1.

This type of problem is important to analyze the stability of a solution or bifurcation point itself in mathematically rigorous sense. By enclosing the triple (u, v, λ) of (4.20), we can obtain an exact solution of nonlinear equation and eigenpair of the operator which was linearized at the exact solution.

Application 3: Eigenvalue problem for non-commutative harmonic oscillators [14]

The purpose of this research is to develop a verified numerical method for computing the eigenvalues and eigenfunctions of the following system:

$$Q_{(\alpha,\beta)} \equiv I_{(\alpha,\beta)} \left(-\frac{1}{2}\frac{d^2}{dx^2} + \frac{x^2}{2} \right) + J \left(x\frac{d}{dx} + \frac{1}{2} \right).$$

Here $x \in \mathbf{R}$ and the matrices $I_{(\alpha,\beta)}$ and J are given by

$$I_{(\alpha,\beta)} \equiv \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix},\ J \equiv \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \in \mathrm{Mat}_2(\mathbf{R}),$$

and α and β are positive real constants satisfying $\alpha\beta > 1$. It is known that $Q_{(\alpha,\beta)}$ defines a self-adjoint positive definite operator, hence has a discrete spectrum. Since the spectrum is defined via non-commuting two matrices $I_{(\alpha,\beta)}$ and J for $\alpha \neq \beta$, the system is called by the non-commutative harmonic oscillator.

In case of $\alpha = \beta$, the eigenvalue is determined as

$$\lambda_n = (n + 1/2)\sqrt{\alpha^2 - 1} \quad (\lambda \in \mathbf{N})$$

by representation theory, but it is very difficult to describe explicitly the eigenstate, in case $\alpha \neq \beta$. In [14] the spectral method using Hermite functions was used together with our enclosure method, and we obtained very accurate enclosure results in case that α is different from β. Moreover we have proved that some enclosed eigenvalues have multiplicity 2. Basically our method cannot be applied to enclose multiple eigenvalues, but in this case we could make use of the

parity of eigenfunctions and monotonicity of eigenvalues. We can say that this application became a good example of *computer assisted proof* in pure mathematics.

Application 4: Kolmogorov problem [15]

We consider the following Navier-Stokes equations

$$\begin{cases} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \nu\Delta u - \frac{1}{\rho}\frac{\partial p}{\partial x} + \gamma\sin\left(\frac{\pi y}{b}\right), \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \nu\Delta v - \frac{1}{\rho}\frac{\partial p}{\partial y}, \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0, \end{cases} \tag{4.21}$$

where (u, v), ρ, p and ν are velocity vector, mass density, pressure and kinematic viscosity, respectively and γ is a constant representing the strength of the sinusoidal outer force. The flow region is a rectangle $[-a, a]\times[-b, b]$ and the periodic boundary condition is imposed in both directions. We define the aspect ratio α as b/a.

We have presented a rigorous theorem which proves the stability of certain solutions by the verified computation. The linearized eigenvalue problem arising in this problem is not self-adjoint and, accordingly, it is quite difficult to treat theoretically.

It is known that nontrivial solutions bifurcate from the basic solution at a certain Reynolds number if and only if $0 < \alpha < 1$. If α is small enough or close to unity, then the stability could be proved mathematically. However, stability in the intermediate range of α is very difficult to prove. We therefore took a new approach to this stability problem by employing the theory of verified computation. Our result shows that the stability is rigorously verified for the cases of $\alpha = 0.4, 0.7$, and 0.8. Our method can be applied, in principle, to any $\alpha \in (0, 1)$.

In [15] we reformulated above problem using a stream function and enclosed an eigenfunction corresponding to the zero eigenvalue as well as the Reynolds number which attain the eigenvalue "zero". Using the results we proved the stability of a bifurcating solution. This is also a good example of computer assisted proof for the problem which is difficult to treat theoretically.

Remark: Our enclosure method proposed in [10, 12] needs the simplicity of the aiming eigenvalues. (Of course such information is not needed in advance.) It means that in principle multiple eigenvalues cannot be enclosed by this method. Concerning the enclosing multiple eigenvalues, see [24]. In [24], for the eigenvalue problem $Lu = \lambda u$, let n be an expected multiplicity of an eigenvalue λ and consider the following system:

$$LY = YM,\ Y \equiv (y_1, ..., y_n),\ M \equiv \begin{pmatrix} m_{11} & \dots & m_{1n} \\ \vdots & \ddots & \vdots \\ m_{n1} & \dots & m_{nn} \end{pmatrix}$$

for $Y_i \in H_0^1(\Omega)$ and $m_{ij} \in \mathbf{R}$. In their mehod the multiple eigenvalue and the basis of corresponding invariant subspace are verified by enclosing a solution $(Y, M) \in (H_0^1(\Omega))^n \times \mathbf{R}^{n^2}$. This method is an extension of the enclosing method for multiple eigenvalues of matrix [22].

5 Essential spectrum problem

Finally we describe our recent challenge to *exclude* eigenvalues in *spectral gaps*, i.e. we treat an operator which has the essential spectrum with gaps.

We consider the following eigenvalue problem

$$Lu \equiv -u'' + q(x)u + s(x)u = \lambda u, \quad x \in \mathbf{R}, \tag{5.1}$$

where we assume that $q(x) \in L^\infty(\mathbf{R})$ is a periodic function and $s \in L^\infty(\mathbf{R})$ satisfies $s(x) \to 0$ $(|x| \to \infty)$. Here we consider the case that $s(x) = ce^{-x^2}$. Our aim is *excluding* an eigenvalue in some interval (hopefully between two essential spectra).

The essential spectrum of L could be obtained as follows.

At first the essential spectrum of the operator

$$L_0 u \equiv -u'' + q(x)u$$

is obtained using the result by Eastham [5].

Theorem 3. *Let $q(x)$ be a periodic function in $(0, r)$ and consider the following two eigenvalue problems:*

I. Periodic eigenvalue problem:

$$\begin{cases} -u'' + q(x)u = \lambda u, \\ u(0) = u(r),\ u'(0) = u'(r), \end{cases} \tag{5.2}$$

and

II. Semi-periodic eigenvalue problem:

$$\begin{cases} -u'' + q(x)u = \mu u, \\ u(0) = -u(r),\ u'(0) = -u'(r). \end{cases} \tag{5.3}$$

Then for each eigenvalues $\{\lambda_n\},\ \{\mu_n\}$ we have

$$\lambda_0 < \mu_0 \le \mu_1 < \lambda_1 \le \lambda_2 < \mu_2 \le \mu_3 \le \cdots, \tag{5.4}$$

and the essential spectra of L_0 are obtained as

$$(\lambda_{2m}, \mu_{2m}),\ (\mu_{2m+1}, \lambda_{2m+1}) \quad m = 0, 1, 2, \cdots. \tag{5.5}$$

Moreover we are able to confirm that L is a compact perturbation of L_0. Therefore essential spectra of L and L_0 coincide. (cf. [26])

We try to exclude eigenvalues of L in spectral gaps by the method proposed in [10] or [12].

We first consider the case that $q(x) = a \cdot \cos(2\pi x)$ for $a \in \mathbf{R}$. Then we obtain (approximate) $\{\mu_i\}$ and $\{\lambda_i\}$ for e.g. $a = 5.0$ as follows:

$$\begin{aligned}\lambda_0(-0.624017) &< \mu_0(7.292924) < \mu_1(12.287917) < \lambda_1(39.425660)\\ &< \lambda_2(40.049607) < \mu_2(88.863540) < \cdots,\end{aligned}$$

The first spectral gap is (μ_0, μ_1) and the second spectral gap is (λ_1, λ_2) which is much narrow than the first spectral gap. Our first target is to exclude eigenvalue in the first spectral gap.

5.1 Fixed Point Formulation

For a real number $\lambda \notin \sigma_{ess}(L_0)$, consider a linear equation

$$(L - \lambda)u = 0 \quad \text{on } \mathbf{R}. \tag{5.6}$$

Since it is clear that (5.6) has a trivial solution $u = 0$, if we validate the uniqueness of the solution of (5.6) by the method described below, it implies that any λ is not an eigenvalue of L.

Since the inverse of $L_0 - \lambda$ exists if $\lambda \notin \sigma_{ess}(L_0)$, we have

$$\begin{aligned}(L - \lambda)u = 0 &\Leftrightarrow (L_0 - \lambda)u + su = 0\\ &\Leftrightarrow u = -(L_0 - \lambda)^{-1}(su).\end{aligned}$$

By Floquet Theory there exist fundamental solutions $\psi_1(x), \psi_2(x)$ of $(L_0 - \lambda)\psi = 0$ s.t.

$$\psi_1(x) = e^{\mu x}p_1(x), \;\; \psi_2(x) = e^{-\mu x}p_2(x), \tag{5.7}$$

where μ is the characteristic exponent and $p_1(x), p_2(x)$ are periodic functions. Using those fundamental solutions we define the Green's function $G(x, y, \lambda)$ [5] for $-\infty < x, y < \infty$ by

$$G(x, y, \lambda) = \begin{cases} \psi_1(x)\psi_2(y)/W(\psi_1, \psi_2)(x) & (x \le y)\\ \psi_2(x)\psi_1(y)/W(\psi_1, \psi_2)(x) & (x \ge y)\end{cases} \tag{5.8}$$

where $W(\psi_1, \psi_2)(x) \equiv \psi_1(x)\psi_2'(x) - \psi_1'(x)\psi_2(x)$ stands for the Wronskian.

Lemma 2. *Above $W(\psi_1, \psi_2)(x)$ is the constant function.*

Proof. We have

$$W(\psi_1, \psi_2)'(x) = \psi_1(x)\psi_2''(x) - \psi_1''(x)\psi_2(x),$$

and this vanishes because of the relation

$$\frac{\psi_1''}{\psi_1} = \frac{\psi_2''}{\psi_2}. \;\blacksquare$$

Therefore we express $W(\psi_1, \psi_2)(x)$ as ξ and rewrite $G(x, y, \lambda)$ as

$$G(x, y, \lambda) = \begin{cases} \psi_1(x)\psi_2(y)/\xi & (x \le y)\\ \psi_2(x)\psi_1(y)/\xi & (x \ge y)\end{cases} \tag{5.9}$$

Using this Green's function we have [5]

$$(L_0 - \lambda)^{-1}f = \int_{\mathbf{R}} G(x, y, \lambda)f(y)dy. \tag{5.10}$$

Using a compact operator

$$F_\lambda u \equiv -\int_{\mathbf{R}} G(x, y, \lambda)s(y)u(y)dy$$

on $H^1(\mathbf{R})$ we have a fixed point equation

$$u = F_\lambda u \tag{5.11}$$

which is equivalent to $(L - \lambda)u = 0$.

5.2 Projection and interpolation

Let $\Omega_M \equiv [-M, M]$ be a bounded interval on $\mathbf{R}$ and set $\widetilde{\Omega}_M \equiv \mathbf{R}\backslash\Omega_M$. For any $v \in H^1(\mathbf{R})$ we consider the following decomposition:

$$v_M(x) \equiv v(x)|_{\Omega_M}, \; \widetilde{v}_M(x) \equiv v(x)|_{\widetilde{\Omega}_M}. \tag{5.12}$$

Defining the projections

$$P_M : H^1(\mathbf{R}) \to H^1(\Omega_M)$$

and

$$\widetilde{P}_M : H^1(\mathbf{R}) \to H^1(\widetilde{\Omega}_M)$$

as $P_M(v) = v_M$ and $\widetilde{P}_M(v) = \widetilde{v}_M$, we decompose (5.11) into the finite and infinite *interval* parts:

$$\begin{cases} P_M u &= P_M F_\lambda(u),\\ \widetilde{P}_M u &= \widetilde{P}_M F_\lambda(u).\end{cases} \tag{5.13}$$

Let Π be the piecewise linear interpolation operator on Ω_M and we further decompose the former part of (5.13) into the finite and infinite *dimensional* parts:

$$\begin{cases} \Pi P_M u &= \Pi P_M F_\lambda(u),\\ (I - \Pi)P_M u &= (I - \Pi)P_M F_\lambda(u).\end{cases} \tag{5.14}$$

Let $S_h(\Omega_M)$ denote the set of continuous and piecewise linear polynomials on Ω_M with uniform mesh $-M = x_0 < x_1 < \cdots < x_N = M$ and mesh size h. Due to Schultz [23] we have the following error estimation for Π:

Lemma 3. *If $f \in PC^{2,\infty}(\Omega_M) \equiv \{\varphi \in C^2(\Omega_M) \mid \|\varphi''\|_\infty < \infty\}$, then we have*

$$\|f - \Pi f\|_\infty \le \frac{1}{8}h^2\|f''\|_\infty. \tag{5.15}$$

5.3 Newton-like method and verification condition

Since we apply a Newton-like method only for the former part of (5.14), we define the following operator:

$$\mathcal{N}_\lambda(u) \equiv Pu - [I - F_\lambda]_M^{-1}(Pu - PF_\lambda(u)),$$

where $P \equiv \Pi P_M$.

Here we assumed that the restriction to $S_h(\Omega_M)$ of the operator $\Pi[I - F_\lambda] : S_h(\Omega_M) \to S_h(\Omega_M)$ has the inverse $[I - F_\lambda]_M^{-1}$. The validity of this assumption can be numerically confirmed in actual computations.

We next define the operator

$$T_\lambda : H^1(\Omega_M) \times H^1(\widetilde{\Omega}_M) \longrightarrow H^1(\Omega_M) \times H^1(\widetilde{\Omega}_M)$$

for $u_M \equiv P_M u$ and $\widetilde{u}_M \equiv \widetilde{P}_M u$ by

$$T_\lambda \begin{pmatrix} u_M \\ \widetilde{u}_M \end{pmatrix} \equiv \begin{pmatrix} \mathcal{N}_\lambda(u) + (I - \Pi)P_M F_\lambda(u) \\ \widetilde{P}_M F_\lambda(u) \end{pmatrix}.$$

Then we have the following equivalence relation

$$\begin{pmatrix} u_M \\ \widetilde{u}_M \end{pmatrix} = T_\lambda \begin{pmatrix} u_M \\ \widetilde{u}_M \end{pmatrix} \Longleftrightarrow u = F_\lambda(u).$$

Our purpose is to find a unique fixed point of T_λ in a certain set $U \subset L^\infty(\Omega_M) \times L^2(\widetilde{\Omega}_M)$, which is called a 'candidate set'. Given positive real numbers γ, α_M and β we define the corresponding candidate set U by

$$U \equiv \begin{pmatrix} U_M + [\alpha_M] \\ [\beta] \end{pmatrix}, \tag{5.16}$$

where

$$U_M \equiv \{v_h \in S_h(\Omega_M) \mid \|v_h\|_{L^\infty(\Omega_M)} \le \gamma\}, \tag{5.17}$$

$$[\alpha_M] \equiv \{v_\perp^M \in (I-\Pi)(H^1(\Omega_M)) \mid \|v_\perp^M\|_{L^\infty(\Omega_M)} \le \alpha_M\}, \tag{5.18}$$

$$[\beta] \equiv \{\widetilde{v} \in H^1(\widetilde{\Omega}_M) \mid \|\widetilde{v}\|_{L^2(\widetilde{\Omega}_M)} \le \beta\}. \tag{5.19}$$

If the relation

$$\overline{T_\lambda(U)} \subset \mathrm{int}(U) \tag{5.20}$$

holds, by the linearity of T_λ, there exists the unique fixed point $u \equiv 0$ of T_λ in U, which implies that λ is not an eigenvalue of L.

Decomposing (5.20) into two types of finite and infinite parts we have a sufficient condition for (5.20) as follows:

$$\sup_{u \in U} \|\mathcal{N}_\lambda(u)\|_{L^\infty(\Omega_M)} < \gamma, \tag{5.21}$$

$$\sup_{u \in U} \|(I - \Pi)P_M F_\lambda(u)\|_{L^\infty(\Omega_M)} < \alpha_M, \tag{5.22}$$

$$\sup_{u \in U} \|\widetilde{P}_M F_\lambda(u)\|_{L^2(\widetilde{\Omega}_M)} < \beta. \tag{5.23}$$

So in oder to construct a suitable set U which satisfies the condition (5.20), we find the positive real numbers γ, α_M and β which satisfy the conditions (5.21)-(5.23).

5.4 Verification for the fundamental solutions

In order to obtain the fundamental solutions ψ_1 and ψ_2 for $(L_0 - \lambda)\psi = 0$, it is sufficient to enclose the functions ϕ_1 and ϕ_2 which are solutions for the following equations:

$$\begin{cases} -\phi_1'' + q\phi_1 - \lambda\phi_1 = 0 \quad \text{in } [0, M] \\ \phi_1(0) = 1,\ \phi_1'(0) = 0 \end{cases} \tag{5.24}$$

$$\begin{cases} -\phi_2'' + q\phi_2 - \lambda\phi_2 = 0 \quad \text{in } [0, M] \\ \phi_2(0) = 0,\ \phi_2'(0) = 1 \end{cases} \tag{5.25}$$

We define

$$V \equiv W^1_{\infty,0}(0, M) \cap \left(\bigwedge_{i=0}^{N} C^1[x_i, x_{i+1}] \right).$$

Setting $\phi_1(M) = \kappa$, $\phi_2(M) = \tau$ and transforming

$$\widetilde{\phi}_1(x) \equiv \phi_1(x) + \frac{1 - \kappa}{M}x - 1,\ \widetilde{\phi}_2(x) \equiv \phi_2(x) - \frac{\tau}{M}x,$$

we consider the following problems:

Find $(\widetilde{\phi}_1, \kappa) \in V \times \mathbf{R}$ **s.t.**

$$\begin{cases} -\widetilde{\phi}_1'' + (q - \lambda)(\widetilde{\phi}_1 + \frac{\kappa - 1}{M}x + 1) = 0 \quad \text{in } [0, M] \\ \widetilde{\phi}_1(0) = \widetilde{\phi}_1(M) = 0 \\ \widetilde{\phi}_1'(0) = \frac{1 - \kappa}{M} \end{cases} \tag{5.26}$$

Find $(\widetilde{\phi}_2, \tau) \in V \times \mathbf{R}$ **s.t.**

$$\begin{cases} -\widetilde{\phi}_2'' + (q - \lambda)(\widetilde{\phi}_2 + \frac{\tau}{M}x) = 0 \quad \text{in } [0, M] \\ \widetilde{\phi}_2(0) = \widetilde{\phi}_2(M) = 0 \\ \widetilde{\phi}_2'(0) = 1 - \frac{\tau}{M} \end{cases} \tag{5.27}$$

After solving these problems we evaluate $\phi_1(M)$ and $\phi_2'(M)$ rigorously. Then we can calculate the real values ρ_1 and ρ_2 which are solutions of the quadratic equation:

$$\rho^2 - \{\phi_1(M) + \phi_2'(M)\}\rho + 1 = 0. \tag{5.28}$$

Note that ρ_1 and ρ_2 are characteristic multipliers for $(L_0 - \lambda)\psi = 0$ and characteristic exponents μ_1 and μ_2 are calculated by the relation $e^{M\mu_i} = \rho_i \ \ (i = 1, 2)$.

Here we mention about the relation between $\phi_1,\ \phi_2$ and $\psi_1,\ \psi_2$. We define the matrix A by

$$A = \begin{pmatrix} \phi_1(M) & \phi_1'(M) \\ \phi_2(M) & \phi_2'(M) \end{pmatrix}.$$

Then clearly ρ_1 and ρ_2 are eigenvalues of A. Let v_1 and v_2 be the corresponding eigenvectors for ρ_1 and ρ_2, respectively. Then we define ψ_1 and ψ_2 by

$$\begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} \equiv \begin{pmatrix} v_1 & v_2 \end{pmatrix}^{-1} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}. \tag{5.29}$$

Now we define p_1 and p_2 by

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \equiv \begin{pmatrix} e^{\mu x}\psi_1 \\ e^{-\mu x}\psi_2 \end{pmatrix}. \tag{5.30}$$

Then we can observe that $p_i(x+M) = p_1(x)$ $(i = 1, 2)$ and the ψ_1 and ψ_2 defined by (5.29) satisfy the relation (5.7).

The actual computation to obtain the set U defined by (5.16) and to enclose the fundamental solutions ψ_1 and ψ_2 will be presented in the forthcoming paper.

References

[1] Bazley, N. W., and Fox, D. W., A Procedure for Estimating Eigenvalues, Journal of Mathematical Physics, 3, No. 3 (1962), 469-471.

[2] Beattie, C., and Goerisch, F., Methods for computing lower bounds to eigenvalues of self-adjoint operators, Numerische Mathematik, 72 (1995), 143-172.

[3] Behnke, H., and Goerisch, F., Inclusions for Eigenvalues of Selfadjoint Problems, In: J.Herzberger(eds.), Topics in Validated Computations-Studies in Computational Mathematics, Elsevier, Amsterdam, 1994.

[4] Chatelin, F., Spectral Approximation of Linear Operators, Academic Press, New York, 1983.

[5] Eastham, M. S. P., The Spectral Theory of Periodic Differential Equations, Scottish Academic Press (1973).

[6] Grisvard, P., Elliptic problems in nonsmooth domains, Pitman Monographs and Survays in Pure and Applied Mathematics 24 (1985), London.

[7] Kato, T., On the upper and lower bounds of eigenvalues, Journal of the Physical Society of Japan, 4 (1949), 334-339.

[8] Kato, T., Perturbation Theory for Linear Operators, Springer, Berlin Heidelberg 1966, 1976.

[9] Křížek, M., Neittaanmäki, P., Finite Element Approximation of Variational Problems and Applications, Longman Scientific and Technical, Harlow (1990).

[10] Nakao, M. T., Yamamoto, N., and Nagatou, K., Numerical Verifications for eigenvalues of second-order elliptic operators, Japan Journal of Industrial and Applied Mathematics, 16, No.3 (1999), 307-320.

[11] Nagatou, K., Yamamoto, N., and Nakao, M. T., An approach to the numerical verification of solutions for nonlinear elliptic problems with local uniqueness, Numerical Functional Analysis and Optimization, 20, 5 & 6 (1999), 543-565.

[12] Nagatou, K., A numerical method to verify the elliptic eigenvalue problems including a uniqueness property, Computing, 63 (1999), 109-130.

[13] Nagatou, K., and Nakao, M. T., An enclosure method of eigenvalues for the elliptic operator linearlized at an exact solution of nonlinear problems, a special issue of Linear Algebra and its Applications on LINEAR ALGEBRA IN SELF-VALIDATING METHODS, 324/1-3 (2001), 81-106.

[14] Nagatou, K., Nakao, M. T., and Wakayama, M., Verified numerical computations for eigenvalues of noncommutative harmonic oscillators, Numerical Functional Analysis and Optimization, 23, 5 & 6 (2002), 633-650.

[15] Nagatou, K., A computer-assisted proof on the stability of the Kolmogorov flows of incompressible viscous fluid, submitted.

[16] Nakao, M.T., A numerical approach to the proof of existence of solutions for elliptic problems, Japan Journal of Applied Mathematics **5** (1988), 313-332.

[17] Nakao, M.T., A numerical approach to the proof of existence of solutions for elliptic problems Ⅱ, Japan Journal of Applied Mathematics **7** (1990), 477-488.

[18] Nakao, M.T. and Yamamoto, N., Self-validating methods (in Japanese), Nihonhyoron-sha, 1998.

[19] Nakao, M.T., Numerical verification methods for solutions of ordinary and partial differential equations, Numerical Functional Analysis and Optimization **22** (3&4) (2001), 321-356.

[20] Plum, M., Eigenvalue inclusions for second-order ordinary differential operators by a numerical homotopy method, Journal of applied mathematics and physics (ZAMP), 41 (1990), 205-226.

[21] Plum, M., Explicit H_2-estimates and pointwise bounds for solutions of second-order elliptic boundary value problems, Journal of Mathematical Analysis and Applications, 165 (1992), 36-61.

[22] Rump, S. M., Computational Numerical Bounds for Multiple or Nearly Multiple Eigenvalues, Linear Algebra and its Applications, 324 (2001), 209-226.

[23] Schultz, M. H., Spline Analysis, Prentice-Hall, London (1973).

[24] Toyonaga, K., Nakao, M. T., and Watanabe, Y., Verified Numerical Computations for Multiple and Nearly Multiple Eigenvalues of Elliptic Operators, Journal of Computational and Applied Mathematics, 147, Issue 1 (2002), 175-190.

[25] Yamamoto, N., and Nakao, M. T., Numerical verifications for solutions to elliptic equations using residual iterations with a high order finite element, Journal of Computational and Applied Mathematics 60 (1995) 271-279.

[26] Yosida, K., Functional Analysis, Springer-Verlag, Berlin (1995).

Interval Tools for ODEs and DAEs

Nedialko S. Nedialkov
Department of Computing and Software
McMaster University, Canada
nedialk@mcmaster.ca

Abstract

We overview the current state of interval methods and software for computing bounds on solutions in initial value problems (IVPs) for ordinary differential equations (ODEs). We introduce the VNODE-LP solver for IVP ODEs, a successor of the author's VNODE package. VNODE-LP is implemented entirely using literate programming. A major goal of the VNODE-LP work is to produce an interval solver such that its correctness can be verified by a human expert, similar to how mathematical results are certified for correctness.

We also discuss the state in computing bounds on solutions in differential algebraic equations.

1. Introduction

We consider the initial-value problem (IVP)

$$y'(t) = f(y), \quad y(t_0) = y_0, \quad y \in \mathbb{R}^n,\ t \in \mathbb{R}. \qquad (1)$$

Since interval methods for IVPs for ordinary differential equations (ODEs) are typically based on Taylor series, which require the computation of Taylor coefficients (TCs) for y up to some order $k \geq 1$, we assume that f is as differentiable as needed.

The initial condition can be in an interval vector $\boldsymbol{y}_0$, that is, $y_0 \in \boldsymbol{y}_0$. If we denote the solution of (1) by $y(t; t_0, y_0)$, we denote by $y(t; t_0, \boldsymbol{y}_0)$ the set of solutions originating from each initial condition in $\boldsymbol{y}_0$:

$$y(t; t_0, \boldsymbol{y}_0) = \big\{ y(t; t_0, y_0) \mid y_0 \in \boldsymbol{y}_0 \big\}.$$

Given $t_{\text{end}} > t_0$, we wish to compute interval vectors that are guaranteed to contain the solution to (1) at points t_j for which $t_0 < t_1 < t_2 < \cdots < t_N = t_{\text{end}}$. Namely, we want to find $\boldsymbol{y}_j$ such that

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_j \quad \text{for all } j.$$

When computing these $\boldsymbol{y}_j$, we use interval methods to enclose roundoff and truncation errors in the computed bounds, thus obtaining rigorous bounds on the true solution of the ODE [24, 44].

When describing the theory of these methods, it is convenient to work with an autonomous ODE. However, this is not a restriction, as the theory can be applied to non-autonomous systems, or a non-autonomous system can be converted into an autonomous one. Also for convenience, we require that $t_{\text{end}} > t_0$, but in general t_{end} may be less than t_0.

Section 2 lists software packages for computing bounds on the solution of (1) and lists various applications. Section 3 outlines some of the theory behind interval methods for IVP ODEs. Section 4 gives an overview of VNODE-LP, elaborates on literate programming (LP), and presents numerical results to illustrate some of the issues in these methods. Section 5 outlines Pryce's [52] structural analysis for solving systems of differential algebraic equations (DAEs) and summarizes work to date on Taylor series methods for DAEs. Conclusions are in the last Section 6.

In this paper, we assume knowledge of interval arithmetic and basic interval techniques (see for example [2, 37]). Intervals, interval vectors, and interval matrices will be in bold font.

2. Software and applications

In Table 1, we list packages for computing bounds on the solution of an IVP ODE. We refer to the methods implemented in the packages AWA [33], ADIODES [54], VNODE [43], VNODE-LP [40], and VODESIA [18] as "traditional" interval methods for IVP ODEs. The COSY VI [9] solver is based on Taylor models [48], and VSPODE [32] can be viewed as a mixture of traditional methods and Taylor models.[1] All require computing TCs, while ValEncIA-IVP [4] needs only the Jacobian of f.

[1] To the author's knowledge, VODESIA is not publicly available, and VSPODE is available by request from the authors.

package	year	language
AWA	1988	Pascal-XSC
ADIODES	1997	C++
COSY VI	1997	Fortran
		C++ interface
VNODE	2001	C++
VODESIA	2003	Fortran-XSC
VSPODE	2005	C++
ValEncIA-IVP	2005	C++
VNODE-LP	2006	C++

Table 1. Packages for computing bounds in IVP ODEs.

A notable contribution to this area are the automatic differentiation (AD) packages FADBAD [7] and TADIFF [8], and now FADBAD++ [55]. They have been instrumental in VNODE, VNODE-LP, and VSPODE for computing TCs of an ODE solution and TCs of the solution to the associated variational equation, and in ValEncIA-IVP for evaluating the Jacobian of f.

In general, traditional methods produce tight bounds on solutions in linear ODEs and in nonlinear problems, when the computed enclosures remain sufficiently small. If the overestimations in the computed bounds start growing, then these bounds typically blow up quickly. In particular, on nonlinear ODEs, if the initial condition set is not very small, or long integration is desired, these methods usually break down because of overestimations on the computed solution sets.

Taylor model integration is more effective than traditional methods at producing tight enclosures on a solution to a nonlinear ODE, with an initial condition set that is not very small and over longer integration intervals. However, Taylor models become expensive computationally on larger problems (more than 5–10 equations), while a traditional method, from the author's experience, can deal with a few hundred equations.

While applications of interval methods for ODEs were scarce ten years ago, we see a variety of applications in the last few years; in particular, after the VNODE solver became available, and COSY VI with its Taylor models became popular.

Application of these methods include rigorous computation of asteroid orbits [11], studying long-term stability of large particle accelerators [12], global optimization for parameter estimation in chemical engineering [31], and simulation of wastewater treatment processes [26]. The VNODE package has been employed in rigorous multibody simulations [3], reliable surface intersection [38, 50], robust evaluation of differential geometry properties [29], computing bounds on eigenvalues [14], parameter and state estimation [25, 53], rigorous shadowing [21, 22], and theoretical computer science [1].

3. Theory

We outline the theory of a traditional interval method for IVP ODEs (Subsection 3.1), discuss briefly Taylor models (Subsection 3.2), and show how they work in VSPODE (Subsection 3.3).

3.1. Traditional methods

Suppose that we have computed $\boldsymbol{y}_j$ at t_j such that

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_j.$$

We advance to the next point in time in two phases.

ALGORITHM I tries to find an interval $[t_j, t_{j+1}]$ and an a priori enclosure $\widetilde{\boldsymbol{y}}_j$ such that $y' = f(y)$, $y(t_j) = y_j$ has a unique solution for all $y_j \in \boldsymbol{y}_j$ and all $t \in [t_j, t_{j+1}]$, and

$$y(t; t_j, \boldsymbol{y}_j) \subseteq \widetilde{\boldsymbol{y}}_j \quad \text{for all } t \in [t_j, t_{j+1}]. \tag{2}$$

Proving existence and uniqueness, and finding $[t_j, t_{j+1}]$ and $\widetilde{\boldsymbol{y}}_j$, is usually based on applying a fixed-point theorem [19, 33, 39]. In practice, if the stepsize $h_{j+1} = t_{j+1} - t_j$ (determined in this phase) is smaller than a value for the smallest allowable stepsize, then normally the integration cannot proceed [40]. That is, we cannot validate existence and uniqueness.

ALGORITHM II uses $\widetilde{\boldsymbol{y}}_j$ to enclose the truncation error of the method and computes a tighter enclosure $\boldsymbol{y}_{j+1}$ at t_{j+1} such that

$$y(t_{j+1}; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_{j+1} \subseteq \widetilde{\boldsymbol{y}}_j. \tag{3}$$

Figure 1 depicts bounds produced in these two phases.[2]

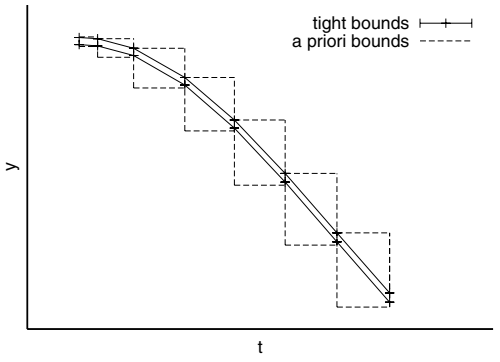


Figure 1. A priori and tight bounds.

Methods for implementing the above two phases are usually based on Taylor series. One reason for their popularity

[2] For this visualization, the tight bounds are connected with lines, which do not necessarily enclose the true solution.

is that it is relatively easy to enclose the truncation error of the method; see also [44]. Before we describe how these two algorithms work, we introduce convenient notation for TCs.

Taylor coefficients. Denote

$$f^{[0]}(y) = y,$$
$$f^{[i]}(y) = \frac{1}{i}\left(\frac{\partial f^{[i-1]}}{\partial y} f\right)(y) \quad \text{for } i \geq 1.$$

Given the IVP $y'(t) = f(y)$, $y(t_j) = y_j$, we have for the ith TC of its solution

$$\frac{y^{(i)}(t_j)}{i!} = f^{[i]}(y_j).$$

Such coefficients can be computed through source-code translation [15] or operator overloading, as in TADIFF and FADBAD++, for example. These packages can compute TCs with a user-supplied data type. In particular, given an interval as an input, they can generate interval TCs.

We note that to compute k coefficients, we require $O(k^2)$ work. Given a stepsize h, one can generate scaled TCs directly, that is $h^i f^{[i]}(y_j)$, instead of computing first $f^{[i]}(y_j)$ and then multiplying it by h^i.

Computing a priori bounds. In VNODE, VNODE-LP, and VSPODE, Algorithm I implements the High-Order Enclosure (HOE) method [45]. It is based on the following result: if y_j is in the interior of $\widetilde{\boldsymbol{y}}_j$, and

$$y_j + \sum_{i=1}^{k-1} (t-t_j)^i f^{[i]}(y_j) + (t-t_j)^k f^{[k]}(\widetilde{\boldsymbol{y}}_j) \subseteq \widetilde{\boldsymbol{y}}_j \quad (4)$$

($k \geq 1$) for all $t \in [t_j, t_{j+1}]$ and all $y_j \in \boldsymbol{y}_j$, then there exists a unique solution to $y' = f(y)$, $y(t_j) = y_j$ for all $y_j \in \boldsymbol{y}_j$ and

$$y(t; t_j, y_j) \in y_j + \sum_{i=1}^{k-1} (t-t_j)^i f^{[i]}(y_j) + (t-t_j)^k f^{[k]}(\widetilde{\boldsymbol{y}}_j)$$

for all $t \in [t_j, t_{j+1}]$ and all $y_j \in \boldsymbol{y}_j$.

When $k = 1$, we obtain the method in AWA, but it restricts the stepsizes similarly to Euler's method. An advantage of the HOE method is that it allows larger stepsizes compared to AWA. Moreover, a good stepsize control can be conveniently incorporated in the HOE method. Details are in [40, 45].

Computing tight bounds. Using $\widetilde{\boldsymbol{y}}_j$, we wish to compute a tighter enclosure $\boldsymbol{y}_{j+1}$ such that (3) holds. A basic approach is to use Taylor series. Writing a Taylor series expansion, we can compute

$$\boldsymbol{y}_{j+1} := \boldsymbol{y}_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(\boldsymbol{y}_j) + h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j)$$

($h_j = t_{j+1} - t_j$), which contains the true solution, but the width of $\boldsymbol{y}_{j+1}$ is

$$\mathrm{w}(\boldsymbol{y}_{j+1}) \geq \mathrm{w}(\boldsymbol{y}_j), \text{ and usually } \mathrm{w}(\boldsymbol{y}_{j+1}) > \mathrm{w}(\boldsymbol{y}_j),$$

even if the solutions are contracting—"naive" method.

To obtain a scheme that could follow contracting solutions, we apply the mean-value evaluation: for any y_j, $\widehat{y}_j \in \boldsymbol{y}_j$,

$$\begin{aligned} & y_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(y_j) + h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j) \\ & \subseteq \widehat{y}_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(\widehat{y}_j) + h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j) \\ & \qquad + \left\{ I + \sum_{i=1}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial y}(\boldsymbol{y}_j) \right\} (\boldsymbol{y}_j - \widehat{y}_j), \end{aligned} \quad (5)$$

where I is the $n \times n$ identity matrix. The above Jacobians can be evaluated through AD by generating TCs for the solution to the associated variational equation [33, 39]. The FADBAD++ [55] package can readily evaluate these Jacobians [40, 43].

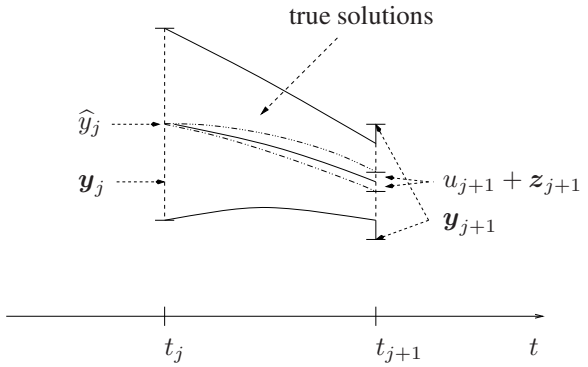


Figure 2. The enclosure at t_{j+1} is formed from an approximate point solution, an enclosure on the local excess, and an enclosure on the propagated global excess.

Based on (5), we can form an enclosure $\boldsymbol{y}_{j+1}$ consisting of (see also Figure 2)
(a) a point approximation $u_{j+1} = \widehat{y}_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(\widehat{y}_j)$;
(b) an enclosure $\boldsymbol{z}_{j+1} = h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j)$ of the truncation error; this enclosure can be viewed as the excess introduced on the current integration step over the true solution set, or *local excess* [39]; and
(c) an enclosure $\boldsymbol{S}_j(\boldsymbol{y}_j - \widehat{y}_j)$, where

$$\boldsymbol{S}_j = I + \sum_{i=1}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial y}(\boldsymbol{y}_j),$$

of the propagated *global excess* [39] to t_{j+1}. One can view $\boldsymbol{y}_j - \widehat{y}_j$ as an enclosure of the global excess at t_j.

Thus,

$$\boldsymbol{y}_{j+1} := u_{j+1} + \boldsymbol{z}_{j+1} + \boldsymbol{S}_j(\boldsymbol{y}_j - \widehat{y}_j). \tag{6}$$

Since we also enclose roundoff errors in the above computations, when executed in machine interval arithmetic, we have a rigorous enclosure on the true solution of (1).

There are two major sources of overestimation in (6). First, we have a high-order Taylor series expansion in time, but only a first-order Taylor series expansion in space. In general, we cannot expect to compute tight bounds for nonlinear ODEs when the initial set, here $\boldsymbol{y}_j$, is not sufficiently small. For linear ODEs, we do not have overestimations in the Jacobian evaluations, as they do not depend on $\boldsymbol{y}_j$.

Second, there is typically wrapping effect [37, 42] originating from the product $\boldsymbol{S}_j(\boldsymbol{y}_j - \widehat{y}_j)$. Because of the wrapping effect, the scheme (6) can follow contracting solutions in a few special cases only [42].

Wrapping effect. For simplicity in the illustration that follows, Figure 3.1, assume $n = 2$, $S_j \in \boldsymbol{S}_j$ is a 2×2 point matrix, and denote $\boldsymbol{a}_j = \boldsymbol{y}_j - \widehat{y}_j$. We are interested in the set $\{S_j a \mid a \in \boldsymbol{a}_j\}$, cf. (6), but we compute in interval arithmetic the box $S_j\boldsymbol{a}_j$, which can introduce significant overestimation over the parallelepiped $\{S_j a \mid a \in \boldsymbol{a}_j\}$. Then, we have to work with this box on the next step, may have another overestimation, and so on. Such overestimations can quickly accumulate, leading to the wrapping effect and a blow up in the computed bounds.

Lohner's QR factorization method [33] for reducing the wrapping effect puts a box in a moving orthogonal coordinate system, and this box matches one of the edges of the enclosed parallelepiped. We can always "match" the longest edge, and intuitively, introduce a smaller overestimation.

Reducing the wrapping effect. Instead of working with box enclosures $\boldsymbol{y}_j$, we can also represent an enclosure on the solution in the form

$$y_j \in \{\, \widehat{y}_j + A_j r_j \mid r_j \in \boldsymbol{r}_j\},$$

where $\widehat{y}_j \in \mathbb{R}^n$, $A_j \in \mathbb{R}^{n\times n}$ is nonsingular, and $\boldsymbol{r}_j \in \mathbb{IR}^n$. ($\mathbb{IR}^n$ denotes the set of n-dimensional interval vectors.)

Instead of computing with (6), we proceed as follows. We set initially

$$A_0 = I, \quad \widehat{y}_0 = \mathrm{m}(\boldsymbol{y}_0), \quad \text{and} \quad \boldsymbol{r}_0 = \boldsymbol{y}_0 - \widehat{y}_0,$$

where $\mathrm{m}(\cdot)$ denotes componentwise midpoint. Then on each step, we find

$$\begin{aligned} \boldsymbol{y}_{j+1} &= u_{j+1} + \boldsymbol{z}_{j+1} + (\boldsymbol{S}_j A_j)\boldsymbol{r}_j, \\ \widehat{y}_{j+1} &= u_{j+1} + \mathrm{m}(\boldsymbol{z}_{j+1}), \end{aligned}$$

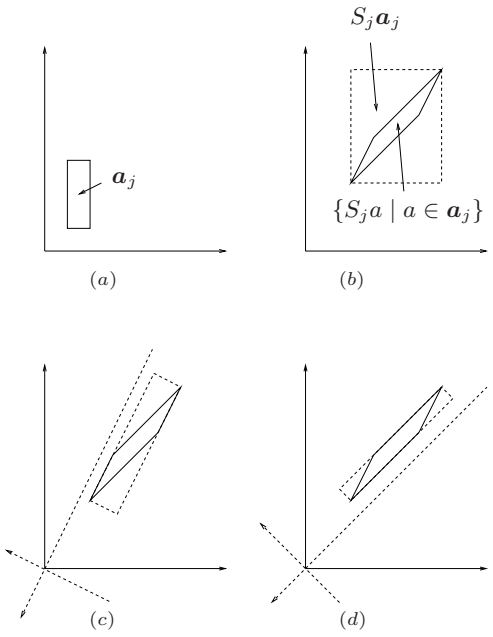


Figure 3. (a) $\boldsymbol{a}_j = \boldsymbol{y}_j - \widehat{y}_j$; (b) wrapping in the original coordinate system; (c) and (d) wrapping in a moving orthogonal coordinate systems that matches one of the edges of the enclosed set.

select a nonsingular A_{j+1}, and form

$$\boldsymbol{r}_{j+1} = \big\{A_{j+1}^{-1}(\boldsymbol{S}_j A_j)\big\}\boldsymbol{r}_j + A_{j+1}^{-1}\big\{\boldsymbol{z}_{j+1} - \mathrm{m}(\boldsymbol{z}_{j+1})\big\}.$$

The selection of A_{j+1} is crucial for the performance of this scheme. If $A_{j+1} = \mathrm{m}(\boldsymbol{S}_j A_j)$, we have the *parallelepiped* method [19, 33]. It frequently breaks down because the A_j become ill conditioned, and the computed bounds become too wide.

In Lohner's QR method, we select $A_{j+1} = Q_{j+1}$ from the QR factorization $Q_{j+1}R_{j+1} = \mathrm{m}(\boldsymbol{S}_j A_j)$. We enclose in a moving orthogonal coordinate system, and we can always "match" the longest edge of the enclosed set by a suitable permutation of the columns of $\mathrm{m}(\boldsymbol{S}_j A_j)$, [33]. An eigenvalue-type analysis shows that the QR method provides good stability for the underlying interval method [42].

3.2. On Taylor models

The above approach uses a parallelepiped to enclose a solution set, which may not be convex. Inherently, methods of this type cannot follow accurately complicated solution sets and can produce large overestimations.

An integration based on Taylor models [10, 35] represents a solution enclosure as a multivariate polynomial in y_0, where $y_0 \in \boldsymbol{y}_0$, plus a small remainder interval. As a

result, such enclosures are not necessarily convex and can describe a solution set much more accurately than a parallelepiped.

Let $\mathcal{F}$ be the set of continuous functions on $\boldsymbol{x} \in \mathbb{IR}^n$ to $\mathbb{R}$, let $p : \mathbb{R}^n \to \mathbb{R}$ be a polynomial of order m, and let $\boldsymbol{r}$ be an interval. A Taylor model is (cf. [48, §2.3])

$$\big\{ f \in \mathcal{F} \mid f(x) \in p(x) + \boldsymbol{r} \ \text{ for all } x \in \boldsymbol{x} \big\}.$$

Arithmetic operations and elementary functions can be implemented on Taylor models such that each of these operations results in a Taylor model [10, 34].

In practice, given a sufficiently differentiable g on $\boldsymbol{x} \in \mathbb{IR}^n$ and $x_0 \in \boldsymbol{x}$, one can apply a (multivariate) Taylor expansion of g around x_0 to construct a polynomial approximation $p(x - x_0)$ to $g(x)$ and enclose the error term (for all $x \in \boldsymbol{x}$) in this expansion in an interval $\boldsymbol{r}$. Then $g(x) \in p(x-x_0)+\boldsymbol{r}$ for all $x \in \boldsymbol{x}$. We refer to $p(x-x_0)+\boldsymbol{r}$ as a Taylor model T_g of g.

Figure 4 illustrates a Taylor model of a function. If its range is enclosed by an interval, we would propagate a box through a computation. With Taylor models, we propagate a Taylor model T_g, a much tighter enclosure of g than an interval enclosure.

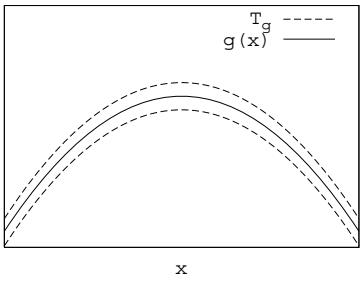


Figure 4. Function and its Taylor model enclosure.

In this paper, we show how Taylor models are incorporated in VSPODE. A good exposition of how "full" Taylor model integration works is in [48].

3.3. Taylor models in VSPODE

Lin and Stadtherr [32] consider the IVP

$$y' = f\big(y(t), \theta\big), \quad y(t_0) = y_0 \in \boldsymbol{y}_0, \quad \theta \in \boldsymbol{\theta}, \qquad (7)$$

where $t \in \mathbb{R}$, $y \in \mathbb{R}^n$, $\boldsymbol{y}_0 \in \mathbb{IR}^n$, $\boldsymbol{\theta} \in \mathbb{IR}^l$, θ is a parameter, and f is assumed sufficiently differentiable, so TCs for y up to some order $k \geq 1$ can be computed.

The goal is to enclose the solution to (7) for all $y_0 \in \boldsymbol{y}_0$ and all $\theta \in \boldsymbol{\theta}$. The method implemented in VSPODE works as follows. Initially, set Taylor models

$$\begin{aligned} y_0 &\in T_{y_0} = \mathrm{m}(\boldsymbol{y}_0) + \big(y_0 - \mathrm{m}(\boldsymbol{y}_0)\big) + [0,0]^n \quad \text{and} \\ \theta &\in T_\theta = \mathrm{m}(\boldsymbol{\theta}) + \big(\theta - \mathrm{m}(\boldsymbol{\theta})\big) + [0,0]^p, \end{aligned}$$

where $[0,0]^n$ denotes the n vector with all components $[0,0]$.

We denote the solution to (7) by $y(t; t_0, y_0, \theta)$. Assume that, at t_j,

$$y(t_j; t_0, y_0, \theta) \in p_j(y_0, \theta) + \boldsymbol{v}_j, \qquad (8)$$

where $p_j : \mathbb{R}^{n+l} \to \mathbb{R}^n$ is a polynomial in y_0 and θ of some degree, say m, and $\boldsymbol{v}_j \in \mathbb{IR}^n$. That is, we have a Taylor model at t_j.

Then at t_{j+1}, for any $y_j \in p_j(y_0, \theta)+\boldsymbol{v}_j$ and any $\theta \in T_\theta$,

$$\begin{aligned} y(t_{j+1}; t_0, y_0, \theta) &\in \sum_{i=0}^{k-1} h_j^i f^{[i]}(y_j, \theta) + h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j, \boldsymbol{\theta}) \\ &\subseteq \sum_{i=0}^{k-1} h_j^i f^{[i]}(p_j + \boldsymbol{v}_j, T_\theta) + \boldsymbol{w}_{j+1}, \end{aligned} \qquad (9)$$

where the $f^{[i]}$ are functions of both y and θ, and

$$\boldsymbol{w}_{j+1} = h_j^k f^{[k]}(\widetilde{\boldsymbol{y}}_j, \boldsymbol{\theta}),$$

in which $\widetilde{\boldsymbol{y}}_j$ is an a priori enclosure over $[t_j, t_{j+1}]$.

The term $\boldsymbol{w}_{j+1}$ is computed with the HOE method. The TCs $f^{[i]}$, for $i = 1, \ldots, k-1$, can be enclosed by performing Taylor model arithmetic through a TC computation. That is, suppose we have a program for computing TCs that works with a generic data type (TADIFF and FADBAD++ allow user-defined types). If we have a Taylor model class with overloaded arithmetic operations and elementary functions, we can execute TC computation with our program and objects of this class. Hence, we can enclose the $f^{[i]}$, and therefore $y(t_{j+1}; t_0, y_0, \theta)$, for all $y_0 \in \boldsymbol{y}_0$ and all $\theta \in \boldsymbol{\theta}$. However, since intervals are involved in evaluating the code list of each $f^{[i]}$ in (9), the widths of the enclosures that we would compute using (9) grow similarly to the widths in the naive Taylor series method described earlier (but likely much slower).

Applying the mean-value theorem to the $f^{[i]}$ and evaluating them with p_j and T_θ, we have

$$\begin{aligned} y(t_{j+1}; t_0, y_0, \theta) \in &\sum_{i=0}^{k-1} h_j^i f^{[i]}(p_j, T_\theta) + \boldsymbol{w}_{j+1} \\ &+ \left(\sum_{i=0}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial y}(\boldsymbol{y}_j, \boldsymbol{\theta}) \right) \boldsymbol{v}_j. \end{aligned} \qquad (10)$$

When evaluating $\sum_{i=0}^{k-1} h_j^i f^{[i]}(p_j, T_\theta)$, we can construct a polynomial $p_{j+1}(y_0, \theta)$ of degree m, enclose the resulting

higher order terms, and include this enclosure and $\boldsymbol{w}_{j+1}$ in an interval (vector) $\boldsymbol{u}_{j+1}$ [32]. That is, we have

$$\sum_{i=0}^{k-1} h_j^i f^{[i]}(p_j, T_\theta) + \boldsymbol{w}_{j+1} \subseteq p_{j+1}(y_0, \theta) + \boldsymbol{u}_{j+1}. \quad (11)$$

Denoting

$$\boldsymbol{V}_j = \sum_{i=0}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial y}(\boldsymbol{y}_j, \boldsymbol{\theta}),$$

we have from (10) and (11) that

$$y(t_{j+1}; t_0, y_0, \theta) \in p_{j+1}(y_0, \theta) + \boldsymbol{u}_{j+1} + \boldsymbol{V}_j \boldsymbol{v}_j. \quad (12)$$

If we implement a scheme based on (12), the product $\boldsymbol{V}_j \boldsymbol{v}_j$ would typically give rise to the wrapping effect. To reduce it, instead of (8), VSPODE uses the representation

$$\{p_j(y_0, \theta) + B_j s \mid y_0 \in \boldsymbol{y}_0, \theta \in \boldsymbol{\theta}, s \in \boldsymbol{s}_j\},$$

where $B_j \in \mathbb{R}^{n \times n}$ is nonsingular, and $\boldsymbol{s}_j \in \mathbb{IR}^n$, as an enclosure on the solution set at t_j. Then (12) becomes

$$y(t_{j+1}; t_0, y_0, \theta) \in p_{j+1}(y_0, \theta) + \boldsymbol{u}_{j+1} + (\boldsymbol{V}_j B_j)\boldsymbol{s}_j.$$

For the next step, B_{j+1} and $\boldsymbol{s}_{j+1}$ are computed as in Lohner's method.

We evaluate Jacobians of $f^{[i]}$ over $\boldsymbol{y}_j$ (and in this case $\boldsymbol{\theta}$) as in traditional methods and deal with the wrapping effect as in Lohner's method. Hence, propagating the global excess in VSPODE is similar to propagating global excess in traditional methods. However, due to the more elaborate enclosures of TCs, this excess often remains smaller (and the enclosures tighter) in VSPODE compared to the excess in traditional methods; see for comparison the numerical results in [32].

4. VNODE-LP

4.1. Motivation

In general, interval methods produce results that can have the power of a mathematical proof. As shown in the previous section, when computing an enclosure of the solution of an IVP ODE, an interval method first proves that there exists a unique solution to the problem and then produces bounds that contain it. When solving a nonlinear equation, an interval method can prove that a region does not contain a solution or compute bounds that contain a unique solution to the problem.

However, if such a method is not implemented correctly, it may not produce rigorous results. Furthermore, we cannot claim mathematical rigor if we miss to include even a single roundoff error in a computation. Therefore, it is of paramount importance to ensure that an interval algorithm is encoded correctly in a programming language.

In the author's opinion, interval software should be produced in a form such that program correctness can be certified in a *human peer-review process*, like a mathematical proof is checked for correctness. This is in contrast to mechanical software verification, when a proof tool is applied to verify code against given specifications.

A major goal of the VNODE-LP work is to implement and document an interval solver for IVPs for ODEs such that its correctness can be verified by a reviewer.

4.2. Literate programming

To accomplish our goal, we have chosen the LP approach. The author has found LP particularly suitable for ensuring that an implementation of a numerical algorithm is a correct translation of its underlying theory into a programming language. With LP, theory, code, and documentation are interwoven in LaTeX-like *web* files.[3] The source code is extracted in a *tangle* process, and the documentation is created in a *weave* process. With VNODE-LP, we have used

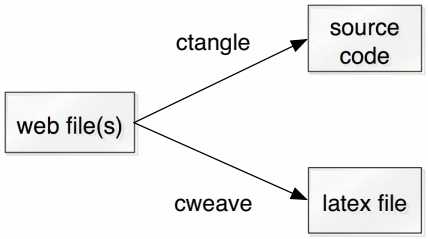


Figure 5. Producing C++ and LaTeX files from web files

the CWEB package [28]. The LP document [40], which contains theory, code, examples, user guide, etc., is generated by executing `cweave` [28] on VNODE-LP's web files; see Figure 5. The C++ code of VNODE-LP and all the examples in [40] are generated by executing `ctangle` [28] on those files (Figure 5).

Some of the benefits of using LP follow.

- We can combine theory, source code, and documentation in a single document.
- With LP, we can produce nearly "one-to-one" translation of the mathematical theory of a method into a computer program. In particular, we can split the theory into small pieces, translate each of them, and keep mathematical expressions and the corresponding code close together in a unified document. This facilitates

[3] "web" is unrelated to the World Wide Web.

verifying the correctness of smaller pieces and of a program as a whole.

- Since theory and implementation are in a single document, it is easier to keep them consistent, compared to having separate theory, source code, and documentation.

Finally, if the correctness of the manuscript [40] is confirmed by reviewers in a peer-review-like process, we may trust in the correctness of the implementation of VNODE-LP, and accept the bounds it computes as rigorous. When claiming rigor, we presume that the operating system, compiler, and the packages VNODE-LP uses do not contain errors.

4.3. Overview

In its basic usage, VNODE-LP attempts to compute bounds on the solution of

$$y' = f(t, y), \quad y(t_0) \in \boldsymbol{y}_0$$

at a given point $t_{\text{end}} \neq t_0$. (Here, $\boldsymbol{y}_0 \in \mathbb{IR}^n$, and $t_0, t_{\text{end}} \in \mathbb{R}$.) If VNODE-LP cannot reach t_{end}, for example if the bounds are too wide, bounds on the solution at some t^* between t_0 and t_{end} are returned.

This package is applicable to ODE problems for which derivatives of the solution exist to some order. Hence, the code list of f should not contain functions such as abs or min.

When integrating from t_0 to t_{end}, VNODE-LP can also return on each step an enclosure $\boldsymbol{y}_j$ on the solution at point t_j such that (3) holds, and an enclosure $\widetilde{\boldsymbol{y}}_j$ on the solution over $[t_j, t_{j+1}]$ such that (2) holds. Examples of how such enclosures are obtained are given in [40].

The HOE method (cf. Section 3) is implemented in Algorithm I, and the interval Hermite-Obreschkoff method [39] is implemented in Algorithm II. (The latter can be viewed as a generalization of a Taylor series method.) VNODE-LP features variable stepsize control and constant order. The stepsize is varied such that an estimate of the size of the local excess per unit step in Algorithm I is below a tolerance. Namely, h_j is selected such that

$$h_j^k \left\| \mathrm{w}(f^{[k]}(\widetilde{\boldsymbol{y}}_j)) \right\| \lessapprox h_j \big(\mathrm{atol} + \mathrm{rtol} \cdot \left\| \boldsymbol{y}_j \right\| \big),$$

where atol and rtol are absolute and relative tolerances, respectively, with default values of 10^{-12}, and $\| \cdot \|$ is the infinity norm.

Typical values for the order can be between 20 and 30 (cf. Subsection 4.5), and a default order is set to 20. There is also improved wrapping effect control compared to VNODE [43] by combining the parallelepiped and QR factorization methods [40].

4.4. Packages and platforms

VNODE-LP compiles with either of the interval arithmetic (IA) packages PROFIL/BIAS [27] or FILIB++ [30]. The interface to an IA package is encapsulated in about 25 short wrapper functions that call functions from it [40]. In principle, one should be able to incorporate a different IA package without major difficulties by implementing these wrapper functions.

The automatic differentiation is done through FADBAD++ [55], and the necessary (non-rigorous) linear algebra is done through LAPACK and BLAS.

To date, VNODE-LP has installed successfully with the GNU C++ compiler as shown below:

IA	OS	Architecture
FILIB++	Linux	x86
	Solaris	Sparc
PROFIL	Linux	x86
	Solaris	Sparc
	Mac OSX	PowerPC
	Windows with Cygwin	x86

4.5. Performance

We report numerical results to illustrate some of the issues in this area. The computations are performed on a 3 GHz dual-core Pentium with 2GB RAM and 4MB L2 cache. The operating system is Linux (Fedora), the compiler is `gcc` version 4.1.1, and the IA package is PROFIL/BIAS. VNODE-LP and the supporting packages are compiled with option `-O2`.

Work versus order

We integrate the Lorenz system

$$\begin{aligned} y_1' &= 10(y_2 - y_1) \\ y_2' &= y_1(28 - y_3) - y_2 \\ y_3' &= y_1 y_2 - 8/3\, y_3 \end{aligned} \tag{13}$$

with

$$y(0) = (15, 15, 36)^T, \quad t \in [0, 20]$$

and $\mathrm{atol} = \mathrm{rtol} = 10^{-7}, 10^{-9}, 10^{-11}$, and 10^{-13}. In Figure 6, we plot the user CPU time (in seconds) taken by VNODE-LP versus the order of the method. As can be seen from this figure, choosing the optimal value for the order is not crucial for the efficiency of the integration: any order around 20 yields good performance. Experience shows that, in general, the value for the order that results in the least amount of work is located in a rather flat minimum.

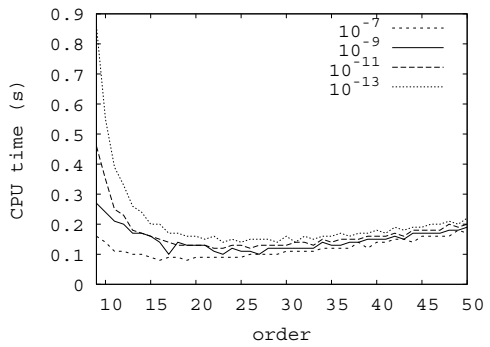


Figure 6. Work versus order for the Lorenz system.

Work versus problem size

We give the DETEST [23] problem C3

$$y' = \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ & & \vdots & & & \\ 0 & \cdots & & 1 & -2 & 1 \\ 0 & \cdots & & 0 & 1 & -2 \end{pmatrix} y$$

with $y(0) = (1, 0, \ldots, 0)^T$ to VNODE-LP.

We integrate with problem sizes $n = 40, 60, \ldots, 300$ for $t \in [0, 5]$. In this and the remaining examples, we use the default order 20 and $\mathrm{atol} = \mathrm{rtol} = 10^{-12}$. In Figure 7, we plot in a log-log scale the CPU time per step versus n; for each n, VNODE-LP takes 8 steps.

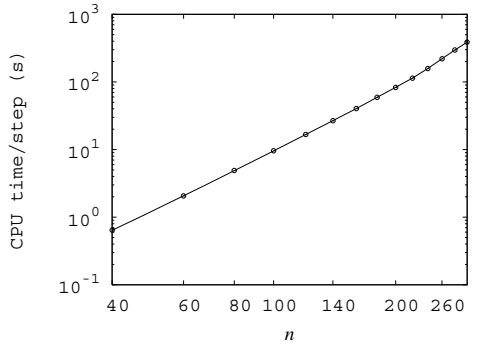


Figure 7. CPU time versus n for the DETEST C3 problem.

On this problem, the linear algebra contributes the most in the total amount of work. From Section 3 (and this plot), it is obvious that this work grows like n^3. The $O(n^3)$ complexity comes from the matrix operations in reducing the wrapping effect, and this complexity is a serious obstacle towards solving larger problems.

Remark. To keep the dependence on an IA package as minimal as possible, the author has implemented the interval linear algebra through the C++ standard template library, not exploiting PROFIL's matrix and vector operations, which are optimized in terms of minimizing rounding mode switches. The present implementation of VNODE-LP does not attempt to minimize the number of these switches in matrix and vector operations. If such optimizations are taken into account, the running time would be reduced, but it will be still $O(n^3)$.

Stiff problems

We integrate Van der Pol's equation (written as a first-order system)

$$y_1' = y_2$$
$$y_2' = \mu(1 - y_1^2)y_2 - y_1$$

with $y(0) = (2, 0)^T$ and $t_{\text{end}} = 200$. We vary μ and report the number of steps and CPU time used by VNODE-LP in Table 2.

μ	steps	CPU time (s)
10^1	2377	0.8
10^2	11697	3.6
10^3	126459	36.1
10^4	1180844	336.4

Table 2. Number of steps and CPU time when integrating the Van Der Pol system.

As μ increases, the stiffness of this problem increases, and VNODE-LP is forced to take very small stepsizes, resulting in an inefficient integration. For an efficient integration of stiff problems, we need a scheme that would allow much larger stepsize, and no such scheme is available to date.

In passing we note that, for the same order of the truncation error, an interval Hermite-Obreschkoff method allows larger stepsizes than an interval Taylor series method [39, 41]. However, as shown in [39], these methods have a restriction on the stepsize due to the associated formula for the truncation error. As a consequence, their stability is determined not only by the stability function of the underlying formula, as in a standard ODE method, but also by the associated formula for the truncation error.

Interval initial conditions

We illustrate how the bounds behave when integrating the Lorenz system with

$$y(0) \in \begin{pmatrix} 15 + [-10^{-4}, 10^{-4}] \\ 15 + [-10^{-4}, 10^{-4}] \\ 36 + [-10^{-4}, 10^{-4}] \end{pmatrix}. \qquad (14)$$

In Figure 14, we plot the bounds on y_1 versus t. In the

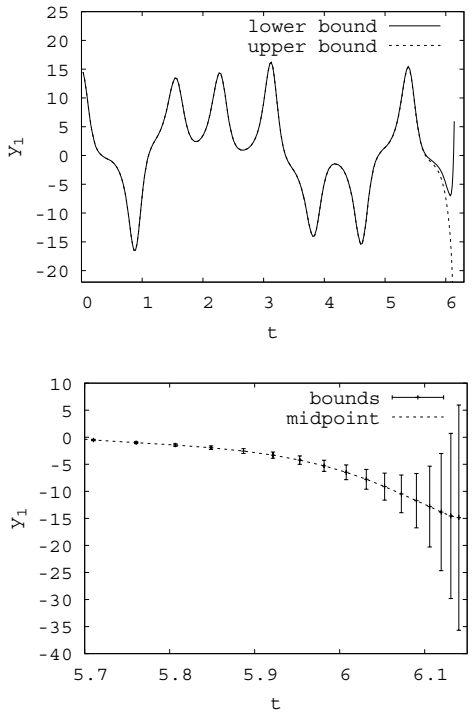


Figure 8. Bounds on y_1 versus t for the Lorenz system (13) with (14).

second plot, we show the computed bounds and their midpoints. Clearly, one should expect a divergence of these bounds. Once they start growing, typically they explode in size very soon. Here, a Taylor model integrator, such as COSY VI, may be able to compute tighter bounds over a longer time interval.

5. On solving DAEs

While several interval solvers for IVPs for ODEs are publicly available, no software is available for computing rigorous bounds on the solution of IVP DAEs. A promising approach for building an interval method (and a solver) for DAEs is Pryce's structural analysis [52] combined with a Taylor series expansion of the solution of a DAE. We outline this analysis and summarize work to date.

5.1. Pryce's structural analysis

We consider an IVP for a DAE system with n equations f_i in n dependent variables $x_j = x_j(t)$. We write informally

$$f_i\big(\, t, \text{ the } x_j \text{ and derivatives of them}\,\big) = 0 \qquad (15)$$

($1 \le i \le n$). The f_i are assumed sufficiently smooth. They can be arbitrary expressions built from the x_j and t using $+, -, \times, \div$, other standard functions, and the differentiation operator d^p/dt^p.

A common measure of the numerical difficulty of a DAE is its *differentiation index* ν_d [13], the number of times the f_i must be differentiated (w.r.t. t) to obtain equations that can be solved to form an ODE system for the x_j. As shown in [46, 47], a method based on Pryce's SA and Taylor series does not find high index inherently hard.

The steps of this SA are summarized below.

1. Form the $n \times n$ signature matrix $\Sigma = (\sigma_{ij})$, where
$$\sigma_{ij} = \begin{cases} \text{order of derivative of } x_j \text{ in } f_i, \text{ or} \\ -\infty \text{ if } x_j \text{ does not occur in } f_i. \end{cases}$$
2. Find a Highest Value Transversal (HVT), which is n positions (i, j) in Σ with one entry in each row and column such that $\sum \sigma_{ij}$ is maximized.
3. Find the smallest "offsets" c_i, $d_j \ge 0$ satisfying
$$\begin{aligned} d_j - c_i &\ge \sigma_{ij} \qquad \text{for all } i, j = 1, \ldots, n \text{ and} \\ d_j - c_i &= \sigma_{ij} \qquad \text{on the HVT.} \end{aligned}$$
Steps 2 and 3 are equivalent to a linear assignment problem and its dual.
4. Form the system Jacobian $\mathbf{J}$, where
$$\mathbf{J}_{ij} = \begin{cases} \dfrac{\partial f_i}{\partial x_j^{(\sigma_{ij})}} & \text{if } d_j - c_i = \sigma_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

Example. Consider the simple pendulum,

$$\begin{aligned} 0 = f &= x'' + x\lambda \\ 0 = g &= y'' + y\lambda - G \\ 0 = h &= x^2 + y^2 - L^2, \end{aligned}$$

which is an index-3 DAE. The dependent variables are x, y, and λ; G (gravity) and L (length) are constants. The signature matrix and the offsets are

$$\Sigma = \begin{array}{c} \\ f \\ g \\ h \\ d_j \end{array} \begin{array}{c} \begin{array}{ccc} x & y & \lambda \end{array} \\ \begin{pmatrix} 2^\circ & -\infty & 0* \\ -\infty & 2* & 0^\circ \\ 0* & 0^\circ & -\infty \end{pmatrix} \\ \begin{array}{ccc} 2 & 2 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 0 \\ 0 \\ 2 \\ \end{array}$$

There are two HVTs, which are marked by $*$ and $\circ$. The system Jacobian is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial x''} & 0 & \frac{\partial f}{\partial \lambda} \\ 0 & \frac{\partial g}{\partial y''} & \frac{\partial g}{\partial \lambda} \\ \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & 0 \end{bmatrix}.$$

If $\mathbf{J}$ is nonsingular at a *consistent point*, then the SA succeeds, and the DAE is solvable in a neighborhood of this point [51, 52]; see also [46].[4] Provided that the SA succeeds, it derives a *structural index*

$$\nu_s = \max_i c_i + \begin{cases} 1 & \text{if some } d_j = 0 \\ 0 & \text{otherwise,} \end{cases}$$

which is the same as that found by the method of Pantelides [49]. It is shown in [52] that $\nu_d \leq \nu_s$; often they are the same.

In the pendulum example, $\mathbf{J}$ is nonsingular for any values of x and y, and the index is $\nu_s = \nu_d = 3$.

To solve (15) by Taylor series, one can use AD to generate functions for evaluating TCs of the equations f_i. Equating these coefficients to zero gives equations that are solved for the TCs of the solution components $x_j(t)$. The offsets prescribe how to organize the computation of TCs [46, 51, 52] for the solution components of (15); that is, what equation to solve and for which TCs of the solution.

We believe that the computation of TCs described in [46] (in the point, approximate case) can be extended to computing interval enclosures of such coefficients. Hence, Algorithm I and Algorithm II could be carried out in the DAE case. A key issue is to ensure that the initial values on the first and subsequent integration steps are consistent with the DAE. That is, they must satisfy the constraints of the DAE, which can include hidden constraints. Pryces's SA identifies the constraints of the DAE using the offsets c_i. Namely,

$$f_i',\ f_i'',\ \ldots, f_i^{(c_i-1)} = 0$$

must hold for all $i = 1, \ldots, n$, from which we can determine $x_j,\ x_j', \ldots, x_j^{(d_j-1)}$ for $j = 1, \ldots, n$.

For example, for the simple pendulum the values for x, x', y, and y' must satisfy the obvious and hidden constraints

$$\begin{aligned} x^2 + y^2 - L^2 = 0 \quad &\text{and} \\ xx' + yy' = 0, \quad & \end{aligned} \tag{16}$$

respectively. In an interval setting, given intervals enclosing x and x', one may apply an interval Newton method to enclose y and y', or given intervals for x, x', y, y', one may verify that they contain a point satisfying (16).

[4] Although applicable to a wide range of DAEs, there are problems on which this SA fails; that is, when $\mathbf{J}$ is singular at a point at which the DAE is solvable. Examples of such problems are discussed in [46, 51, 52].

5.2. Work to date

Chang and Corliss [15] show how to generate Taylor series for the simple pendulum DAE. Then Corliss and Lodwick [17] show how to use interval techniques to obtain bounds on the solution of a simple linear DAE with AWA. They assume that consistent initial conditions are given, and consistency on subsequent steps is a result of AWA's validation algorithm. In [16], they investigate the role of constraints in an interval method for DAEs when applied to the simple pendulum, and in particular, how to use these constraints to verify consistency of user-supplied initial conditions; to suggest consistent initial conditions if necessary; and to tighten initial conditions (on first and subsequent steps) by a Gauss-Seidel iteration or intersecting with the constraints.

Hoefkens [14] uses Pryce's structural analysis to convert a DAE into a generalized ODE, which is then solved in a rigorous way using Taylor models and the COSY package [5]. We note that transforming a DAE into an ODE usually increases the size of the problem to be solved and may destroy its original sparsity pattern.

The rest of this summary includes work on computing approximate DAE solutions using Taylor series. The author has developed a C++ package DAETS for solving high-index, fully-implicit, arbitrary order DAEs in the form (15). This package takes a C++ description of the DAE, generates Σ through operator overloading, finds the problem offsets, and computes TCs using FADBAD++. Then an approximate solution is obtained by summing these coefficients (with appropriate stepsize) and projecting it to satisfy the constraints of the DAE. Theory, algorithmic details, and examples produced by DAETS are given in [46, 47].

Walther and Griewank [20] report of a similar implementation (to DAETS) of a Taylor series method based on Pryce's analysis, but using the ADOL-C package.

Finally, Barrio [5] uses MATHEMATICA to compute Σ, set up an ODE system, and then generate FORTRAN 77 code for evaluating TCs for the ODE system. In [6], he studies the applicability of Taylor series methods for sensitivity analysis of ODEs and DAEs, where DAEs are solved using Pryce's SA and very high-order Taylor series.

6. Conclusion

In the area of interval methods for IVPs for ODEs, we would like to be able to compute efficiently tight bounds on solutions of much larger problems than the current tools can handle. A major obstacle is the $O(n^3)$ complexity, when dealing with the wrapping effect on each integration step. Beating this complexity in a general method may be difficult, but one may develop more efficient methods for classes

of ODEs. For example, the wrapping effect does not occur when integrating quasi-isotone problems [44]. The DETEST C3 (Subsection 4.5) is such a problem, but it was integrated with a general-purpose ODE solver, which does not take into account quasi-isotonicity.

Although high-order Taylor series may be reasonably efficient for mildly stiff ODEs, we do not have an interval method suitable for stiff ODEs. A major challenge and opportunity in this area is to devise an efficient interval method for stiff problems.

The DETEST test set [23] and now the Test Set for IVP Solvers [36] are standard in assessing and comparing (approximate) IVP ODE solvers. As the area of interval ODE solving is maturing, and various interval solvers for IVPs ODEs are available, we need a sound and comprehensive methodology for assessing and comparing these solvers.

Finally, building an interval DAE solver of the quality of existing interval ODE solvers is a challenge, but feasible with the recent progress on both theory and implementation of Taylor series methods for DAEs.

Acknowledgments. George Corliss and John Pryce provided insightful comments, which helped to improve this paper. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] D. Achlioptas. Setting 2 variables at a time yields a new lower bound for random 3-SAT. Technical Report MSR-TR-99-96, Microsoft Research, Microsoft Corp., One Microsoft Way, Redmond, WA 98052, December 1999.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[3] E. Auer, A. Kecskeméthy, M. Tändl, and H. Traczinski. Interval algorithms in modelling of multibody systems. In *Numerical Software with Result Verification*, volume 2991 of *LNCS*, pages 132–159. Springer-Verlag, 2004.

[4] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated modeling of mechanical systems with SmartMOBILE: Improvement of performance by ValEncIA-IVP. In *Reliable Implementation of Real Number Algorithms: Theory and Practice*. Springer-Verlag, to appear.

[5] R. Barrio. Performance of the Taylor series method for ODEs/DAEs. *Appl. Math. Comp.*, 163:525–545, 2005.

[6] R. Barrio. Sensitivity analysis of ODES/DAES using the Taylor series method. *SIAM J. Sci. Comput.*, 27:929–1947, 2006.

[7] C. Bendsten and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods. Technical Report 1996-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, August 1996.

[8] C. Bendsten and O. Stauning. TADIFF, a flexible C++ package for automatic differentiation using Taylor series. Technical Report 1997-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, April 1997.

[9] M. Berz. COSY INFINITY version 8 reference manual. Technical Report MSUCL–1088, National Superconducting Cyclotron Lab., Michigan State University, East Lansing, Mich., 1997.

[10] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361–369, 1998.

[11] M. Berz, K. Makino, and J. Hoefkens. Verified integration of dynamics in the solar system. *Nonlinear Analysis: Theory, Methods & Applications*, 47:179–190, 2001.

[12] M. Berz, K. Makino, and Y.-K. Kim. Long-term stability of the tevatron by verified global optimization. *Nuclear Instruments and Methods*, A558:1–10, 2005.

[13] K. Brenan, S. Campbell, and L. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, second edition, 1996.

[14] B. M. Brown, M. Langer, M. Marletta, C. Tretter, and M. Wagenhofer. Eigenvalue bounds for the singular Sturm-Liouville problem with a complex potential. *J. Phys. A: Math. Gen.*, 36(13):3773–3787, April 2003.

[15] Y. F. Chang and G. F. Corliss. ATOMFT: Solving ODEs and DAEs using Taylor series. *Comp. Math. Appl.*, 28:209–233, 1994.

[16] G. F. Corliss and W. Lodwick. Role of constraints in the validated solution of DAEs. Technical Report 430, Marquette University, Department of Mathematics, Statistics, and Computer Science, Milwaukee, Wisc., March 1996.

[17] G. F. Corliss and W. A. Lodwick. Correct computation of solutions of differential algebraic control equations. *Zeitschrift für Angewandte Mathematik und Mechanik, special issue Numerical Analysis, Scientific Computing, and Computer Science*, pages 37–40, 1996.

[18] S. Dietich. *Adaptive verifizierte Lösung gewöhnlicher Differentialgleichungen*. PhD thesis, University of Karlsruhe, Karlsruhe, Germany, February 2003.

[19] P. Eijgenraam. *The Solution of Initial Value Problems Using Interval Arithmetic*. Mathematical Centre Tracts No. 144. Stichting Mathematisch Centrum, Amsterdam, 1981.

[20] A. Griewank and A. Walther. On the efficient generation of Taylor expansions for DAE solutions by automatic differentiation. In J. Dongarra, K. Madsen, and J. Wasniewski, editors, *PARA'04, State-of-the-art in scientific computing*, volume 3732 of *LNCS*, pages 1103–1111. Springer-Verlag, 2006.

[21] W. Hayes. *Rigorous shadowing of numerical solutions of ordinary differential equations by containment*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 2001.

[22] W. Hayes and K. R. Jackson. Rigorous shadowing of numerical solutions of ordinary differential equations by containment. *SIAM J. Numer. Anal.*, 42(5):1948–1973, 2003.

[23] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM J. Numer. Anal.*, 9(4):603–637, December 1972.

[24] K. R. Jackson and N. S. Nedialkov. Some recent advances in validated methods for IVPs for ODEs. *Appl. Numer. Math.*, 42:269–284, August 2002.

[25] M. Kieffer and E. Walter. Nonlinear parameter and state estimation for cooperative systems in a bounded-error context. In *Numerical Software with Result Verification*, volume 2991 of *LNCS*, pages 107–123. Springer-Verlag, 2004.

[26] M. Kletting, A. Rauh, H. Aschemann, and E. Hofer. Consistency tests in guaranteed simulation of nonlinear uncertain systems with application to an activated sludge process. *J. Comput. Appl. Math.*, 199(2):213–219, 2007.

[27] O. Knüppel. PROFIL/BIAS – a fast interval library. *Computing*, 53(3–4):277–287, 1994.

[28] D. E. Knuth and S. Levy. *The CWEB System of Structured Documentation*. Addison-Wesley, Reading, Massachusetts, 1993.

[29] C. kuo Lee. Robust evaluation of differential geometry properties using interval arithmetic techniques. Master's thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, May 2005.

[30] M. Lerch, G. Tischler, and J. Wolf von Gudenberg. FILIB++—interval library specification and reference manual. Technical Report 279, Universität Würzburg, Germany, 2001.

[31] Y. Lin and M. A. Stadtherr. Deterministic global optimization for parameter estimation of dynamical systems. *Ind. Eng. Chem. Res.*, 2006. in press.

[32] Y. Lin and M. A. Stadtherr. Validated solution of initial value problems for ODEs with interval parameters. In R. L. Muhanna and R. L. Mullen, editors, *Proceedings of 2nd NSF Workshop on Reliable Engineering Computing*, Savannah, GA, February 2006.

[33] R. J. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs– und Randwertaufgaben und Anwendungen*. PhD thesis, Universität Karlsruhe, 1988.

[34] K. Makino and M. Berz. Remainder differential algebras and their applications. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 63–74. SIAM, Philadelphia, Penn., 1996.

[35] K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based validated integrators. Technical Report MSU HEP 40910, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA, 2004.

[36] F. Mazzia and F. Iavernaro. Test set for initial value problem solvers. Technical Report 40, Department of Mathematics, University of Bari, Italy, 2003. `http://pitagora.dm.uniba.it/~testset/`.

[37] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1966.

[38] H. Mukundan, K. H. Ko, T. Maekawa, T. Sakkalis, and N. M. Patrikalakis. Tracing surface intersections with a validated ODE system solver. In G. Elber and G. Taubin, editors, *Proceedings of the Ninth EG/ACM Symposium on Solid Modeling and Applications*. Eurographics Press, June 2004, June 2004.

[39] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, M5S 3G4, February 1999.

[40] N. S. Nedialkov. VNODE-LP — a validated solver for initial value problems in ordinary differential equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, Hamilton, Canada, L8S 4K1, July 2006. VNODE-LP is available at `www.cas.mcmaster.ca/~nedialk/vnodelp/`.

[41] N. S. Nedialkov and K. R. Jackson. An interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. *Reliable Computing*, 5(3):289–310, 1999. Also in T. Csendes, editor, *Developments in Reliable Computing*, pp. 289–310, Kluwer, Dordrecht, Netherlands, 1999.

[42] N. S. Nedialkov and K. R. Jackson. A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations. In A. Facius, U. Kulisch, and R. Lohner, editors, *Perspectives on Enclosure Methods*, pages 219–264. Springer-Verlag, Vienna, 2001.

[43] N. S. Nedialkov and K. R. Jackson. The design and implementation of a validated object-oriented solver for IVPs for ODEs. Technical Report 6, Software Quality Research Laboratory, Department of Computing and Software, McMaster University, Hamilton, Canada, L8S 4K1, 2002.

[44] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Appl. Math. Comp.*, 105(1):21–68, 1999.

[45] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001.

[46] N. S. Nedialkov and J. D. Pryce. Solving differential-algebraic equations by Taylor series (I): Computing Taylor coefficients. *BIT*, 45:561–591, 2005.

[47] N. S. Nedialkov and J. D. Pryce. Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian. *BIT*, 2007. To appear.

[48] M. Neher, K. R. Jackson, and N. S. Nedialkov. On Taylor model based integration of ODEs. *SIAM J. Numer. Anal.*, 45:236–262, 2007.

[49] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM. J. Sci. Stat. Comput.*, 9:213–231, 1988.

[50] N. M. Patrikalakis, T. Maekawa, K. H. Ko, and H. Mukundan. Surface to surface intersection. In L. Piegl, editor, *International CAD Conference and Exhibition, CAD'04*, Thailand, May 2004.

[51] J. D. Pryce. Solving high-index DAEs by Taylor Series. *Numerical Algorithms*, 19:195–211, 1998.

[52] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

[53] N. Ramdani, N. Meslem, T. Raïssi, and Y. Candau. Set-membership identification of continuous-time systems. 14th IFAC Symposium on System Identification, Newcastle, Australia, 2006.

[54] O. Stauning. *Automatic Validation of Numerical Solutions*. PhD thesis, Technical University of Denmark, DK-2800, Lyngby, Denmark, October 1997.

[55] O. Stauning and C. Bendtsen. FADBAD++ web page, May 2003. FADBAD++ is available at `www.imm.dtu.dk/fadbad.html`.

Computer-assisted proofs

Arnold Neumaier
Fakultät für Mathematik, Universität Wien
Nordbergstr. 15, A-1090 Wien, Austria
Arnold.Neumaier@univie.ac.at
http://www.mat.univie.ac.at/∼neum/

Abstract

This paper discusses the problem what makes a computer-assisted proof trustworthy, the quest for an algorithmic support system for computer-assisted proof, relations to global optimization, an analysis of some recent proofs, and some current challenges which appear to be amenable to a computer-assisted treatment.

1 Introduction

Computer-assisted proofs have a long history. Perhaps one of the oldest computer-assisted proofs dates back to the year 1610:

Theorem (Ludolf van Ceulen 1610)
3.14159 26535 89793 23846 26433 83279 26433 deviates from π by less than 10^{-35}.

Proof. Apply the algorithm of Archimedes until the error is small enough. □

At that time, computers were human, slow and not very reliable. The running time on the biological computer quoted as the author of the theorem was over 14 years; the amount of external storage (paper) used is not recorded.

The algorithm of Archimedes was well-known since antiquity. But nobody checked the correctness of van Ceulen's implementation of it, nor inspected anyone the logfiles produced (probably they were not even kept as record). Nevertheless, the theorem is correct and was always regarded as correct.

Improved error bounds, using a much faster algorithm, were computed (on a similar computer) by SHANKS 1893. Shanks approximated π to 707 decimal digits. Again, nobody checked the correctness of the proof or the implementation. However, this time, although based on an algorithm known to be correct, the theorem turned out – much later – to be false (FERGUSON 1945): Only the first 527 digits (and a few random later digits) were correct. It didn't matter – nobody needed (or needs today) so many digits of π. But it is a scientific challenge to get things 100% right – even things whose only use is to satisfy the human curiosity. Today, over 10^9 digits of π are believed to be known.

Trusting computations. All our communication, and all our knowledge, is based on trust. Trust in the sources, trust in the tools used to create and/or check statements, trust in one's memory and in one's reasoning power. And trust in computations, if these play a role.

Trust is important in all proofs, not only the computer-assisted ones. We trust the referees, the experts in the fields, the quoted references, or our own expertise. Sometimes, the trust turns out to be unjustified later. Nevertheless, without trust there is no knowledge, for everything can be doubted. In particular, we trust computations

- if the algorithms on which they are based are understandable and have understandable correctness proofs,
- and if they are carried out with a degree of care that – based on a finite amount of past experience on similar computations – we learnt to rate as reliable.

In case of doubt, we may want to have independent implementations with which the computations can be done. Preferably these implementations should be based upon different algorithms to increase our confidence in the correctness.

The Pentium bug a few years ago shows that computer hardware cannot be trusted unconditionally. Nevertheless, we expect that significant bugs are sooner or later found and then corrected. The search for bugs usually begins with getting an unexpected contradiction to supposed behavior, followed by debugging – the search for the piece in the long chain of arguments or instructions which had no true justification. After the corrections everything depending on the correction must be repeated. For mistakes in crucial places (such as in Shank's calculation of π) on which many other things depend, a large part of the results may be wrong.

On the other hand, in robust, well-structured mathematical proofs, there are many connections between the concepts and arguments used. This has the consequence that most results remain correct when a particular link is found wanting. For example, Russell's paradox invalidated the logical basis of set theory, but hardly affected the bulk of mathematics. This explains that mathematics as a whole thrives, in spite of many wrong published theorems. But it shows the importance of plausibility checks and/or formal verification tools.

Proof checking. Checking a proof ist often (not always) much easier than finding it. Finding the proof of the following result took at the time about 150 days of (human) computer time. Checking it was a matter of 20 minutes.

Theorem. (Cole 1903)
$2^{67} - 1$ is not a prime.

Proof. $2^{67} - 1 = 761838257287 * 193707721.$ □

Most mathematicians regard this as a perfectly valid proof, although the details are long and tedious, and error prone (but doable) for manual checking. But the underlying algorithm is familiar to many people (i.e., for the present purposes, biological computers with independent implementations of the same algorithm) so that trust is granted easily.

The traditional check of numerical calculations with a pocket calculator happens to give the same results for the left and right side of the equation in the proof. We know, however, that pocket calculators are not reliable, and that rational multiprecision arithmetic or interval arithmetic are needed to guard against arithmetical errors in computer calculations.

No one ever will probably want to check the details of a proof of the following theorem, proved (using the formal proof system Coq) by long computations:

Theorem. (CAPROTTI & OOSTDIJK [1])
The 40 decimal digit number

90262 58083 38499 68604 49366 07214 23078 01963

is a prime.

Thus having reduced a mathematical problem to algorithmic computations together with a credible execution of the computations is generally considered enough evidence to count as a proof – except when the algorithms employed are unfamiliar.

When the four-color-theorem was first proved in 1976, the heavy computational part (weeks of combinatorial computations) created strong sentiments about whether it was a valid proof. In the mean time, reservations largely subsided. The computational part of the four-color-problem has become a test problem in mixed-integer linear programming, (with 1372 binary variables and 4944 linear inequality constraints). See FERRIS et al. [3]. Although still nontrivial to solve, its mathematical complexity is now embedded in a well-tested algorithmic framework with many independently written computer codes. Thus it is tamed, understood, even though the details are executed by a computer.

The key for routine acceptability is therefore to reduce a problem to one which can be settled by computer using general purpose software available to many and used by many.

2 The structure of computer-assisted proofs

Typically, computer-assisted proofs which don't consist of pure computation proceed in three stages, which are not always clearly separated:

Stage 1. Qualitative conceptual understanding reduces the problem to a finite number of individual subproblems, each characterized by a small number of parameters.

Stage 2. The analysis of each subproblem results in a number of equations and/or inequalities satisfied for a counterexample.

Stage 3. A specialized computational algorithm is executed, showing that the resulting constraint satisfaction problems have no solution.

The third stage is completely independent of the remainder of the proof. But the formulation of the constraints that make the third stage computationally tractable may involve lots of trial and error.

This leads to the challenge to create software which handles the third step stage automatically and reliably, without the need on the user's side to understand the details. This will allow nonexpert users to pose the computational part of a desired proof in a straightforward (close to) mathematical language, and then run the verification software to check whether the wanted statement can in fact be algorithmically verified.

Such a software system would relieve users from having to think about how to do the computations, so that they can concentrate on the formulation of the constraints. If the software is fast enough, it would allow users to try many formulations and select the the most successful one.

In addition, this would allow computer-assisted proofs to become much more transparent than at present (where the computational and the conceptual parts of the proofs are typically intermingled). It would also make independent checks much easier since the statements which are to be checked by computation are cleanly separated from the conceptual arguments.

Computer-assisted proofs in analysis always require interval methods; see, e.g., [4, 10, 12]. But the typical user of such a verification system should need to know as little about the intricacies of reducing overestimation in interval computations as the typical player of a computer game knows about the underlying computational geometry.

The success of Matlab as an algorithmic language for approximate numerical calculations is largely due to the simplicity with which mathematical algorithms can be encoded. The success of LaTeX as a mathematical typesetting language is largely due to the flexibility with which traditional mathematical notation can be encoded. Therefore, the part of the verification software which the user sees should be as simple to use as a Matlab editor and flexible enough that not much more than elementary LaTeX knowledge is needed to use the software.

The double bubble proof. To see what such a system must be able to solve, I made a study of the computer-assisted proof of the double-bubble conjecture by HASS et al. [6]. From the perspective of the interval community, a short synopsis of the problem and its solution was given by Andreas FROMMER [4]. It is not easy to separate the computationally proved part into a mathematical statement involving only the conditions to be checked without further reference to the geometric sources. Such a separation is needed, however, to enable experts in computer verification but not in geometry to make an independent check of the computational part of the proof.

Here is how I understand what they proved by computer, and what together with the conceptual arguments given in their paper suffices to establish the validity of the double bubble conjecture. (Their notation translates to my notation as $y_1 = s_1, y_2 = s_2$.)

Theorem.
The following conditions are inconsistent:

$$c_1^2 + s_1^2 = c_2^2 + s_2^2 = 1,$$

$$s_1 \geq 0, \;\; s_2 \geq 0, \;\; c_1 \geq 0, \;\; |c_2| \leq 0.5,$$

$$-f_i = f_0 = us_1^2 + s_1c_1\sqrt{3} = us_2^2 + s_2c_2\sqrt{3},$$

$$h_0 = 1 + u, \;\; h_i = 1 - u, \;\; u \in [-1, 9],$$

$$\int_{y_1}^{y_2} \frac{t_0(y)}{\sqrt{4y^2 - t_0(y)^2}}|dy| = \int_{y_1}^{y_2} \frac{t_i(y)}{\sqrt{4y^2 - t_i(y)^2}}|dy|$$

$$\int_{y_1}^{y_2} \frac{y^2t_0(y)}{\sqrt{4y^2 - t_0(y)^2}}|dy| = \int_{y_1}^{y_2} \frac{y^2t_i(y)}{\sqrt{4y^2 - t_i(y)^2}}|dy|$$

where

$$t_0(y) = h_0y^2 - f_0, \;\; t_i(y) = h_iy^2 - f_i.$$

Here the absolute values indicate that the integral is to be understood as a line integral along a path from y_1 to y_2 that reverts its direction at points where the square root vanishes. (It is not difficult to see that this implies that the singularity is harmless.)

Clearly, this theorem needs no geometric knowledge for its proof. It is also nearly obvious (at least for those knowledgeable in interval techniques) that this theorem, if true, can be proved in a finite computation using interval techniques, by implementing a branch and bound process that computes enclosures for both sides of each equation, and discards a box if the resulting intervals are disjoint.

Assuming the theorem to be correct, the success of the method is guaranteed if the enclosures have overestimations that tend to zero with the box size, and this is easy to achieve. The actual implementation of Hass and Schlafly indeed uses only the simplest tools (Riemann sums, interval evaluations, and a limited form of constraint propagation), and finishes the work after having considered 15016 boxes. (More sophisticated strategies would probably shorten the computer-assisted part of the proof considerably.)

Note that formulating the theorem (and adding the above comments) makes the whole proof much more transparent than the proof as actually given. Every reader can see what is the geometric and what is the algebraic part of the proof. Thus, geometers could content themselves checking the geometric part and just trust the programs to settle the algebraic part; the numerical analysts could trust the geometry and check the algebra. In this way the checking work is naturally divided, and the proof easier to check. This adds to the credibility and thus to its trustworthiness. At present one needs to know both sides of the expertise to be able to check the proof.

Had a reliable software system been widely available that allows to check proposed theorems of the above form, the authors would probably have used it – or would have been encouraged by the referees to use it –, to the benefit of all. The availability of such a verification system would make people elsewhere aware of the possibilities of automatic verification, and would give easy access to powerful techniques which are currently for many potential users still a big hurdle.

3 Relations to global optimization

In practice, one often wants to assert something specific about concrete problems in analysis (as opposed to general theorems about all objects of a certain type) To do this rigorously requires the ability to reason with constructively given elements or sets in $\mathbb{R}$, $\mathbb{R}^n$, or some function space. These are typically defined in terms of inequalities (e.g., balls as sets of points x with $\|x - x_0\| \leq r$). Thus we need to be able to check whether certain equations and/or inequalities imply others. Thus we need the ability to work efficiently with equations and inequalities. The assertion

$$A(x) <= B(x) \quad \Rightarrow \quad f(x) < g(x)$$

for vector-valued functions A, B with componentwise inequality and real-valued functions f, g is equivalent with the assertion that the system of inequalities

$$\begin{pmatrix} B(x) - A(x) \\ f(x) - g(x) \end{pmatrix} \geq 0$$

has no solution.

Thus we need to be able to verify rigorously the nonexistence of solutions of systems of inequalities. *Traditional numerical software systems cannot do this – not even approximately – in the absence of special properties such as linearity or convexity.* (Traditional nonlinear programming software just spends some time trying to find a solution and then gives up without a conclusion.) Instead, one needs techniques based on interval arithmetic [10].

The problems stated are semilocal or even global in nature: They must be solved with quantifying over sets with substantial volume (the semilocal case) or even over unbounded sets (the global case).

Finding *best* inequalities leads to global optimization problems. For polynomial optimization problems, proofs of global optimality, and hence of the validity of inequalities, are possible due to necessary and sufficient global optimality conditions recently derived by NEUMAIER & SCHICHL [14]. The proof is based on a deep result from real algebraic geometry, the so-called Positivstellensatz. This, in turn, can be proved using nontrivial techniques from model theory.

The conditions lead to a certificate of optimality, consisting in the coefficients of a polynomial equation. The difficulty of proof finding is reflected in the potentially exponential size of the equation; but in practice often a small certificate exists.

GloptLab [2] is a Matlab implementation of a branch-and-bound method currently under development at the University of Vienna, mainly being implemented by Ferenc Domes. It currently solves quadratic constraint satisfaction problems defined by bound constraints and linear and quadratic equations and inequalities. Every algebraic system of equations and inequalities can be brought into this form, but integrals (such as those needed in the double bubble proof) are not covered at present. GloptLab is based on constraint propagation combined with the maximal exploitation of quadratic (hence small) certificates of infeasibility related to the above results.

Theorem.
For every positive semidefinite matrix G and every matrix $Z \leq 0$, the polynomial $p(x)$ defined by

$$\begin{pmatrix} 1 \\ x \end{pmatrix}^T G \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ \overline{x} - x \\ x - \underline{x} \end{pmatrix}^T Z \begin{pmatrix} 1 \\ \overline{x} - x \\ x - \underline{x} \end{pmatrix} + z^T Q(x) + p(x)$$

satisfies $0 \leq p(x)$ on the feasible set.

In place of longer polynomial certificates, case distinctions (branching steps) are used. Compared with other solvers for global optimization and constraint satisfaction, the emphasis is on mathematical rigor and short proofs, i.e., a small number of branching steps. The implementation uses the Intlab package (RUMP [11]) for interval calculations, the semidefinite programming package SeDuMi [15] for finding certificates, and converters from the COCONUT environment (SCHICHL [13]) to solve global optimization problems posed in the modeling language AMPL.

Rounding error control is essential to make the computations mathematically rigorous. For example, a strictly convex, quadratic inequality constraint defines an ellipsoid whose interval hull is easy to compute analytically. But to cope efficiently with rounding errors is nontrivial. GloptLab uses an algorithm which computes a directed Cholesky factor for a symmetric, positive definite interval matrix $\mathbf{A}$, i.e., a nonsingular triangular matrix R such that $A - R^T R$ is positive semidefinite for all $A \in \mathbf{A}$, and tiny if the components of $\mathbf{A}$ are narrow intervals. In particular, $x^T A x \geq \|Rx\|_2^2$, with little overestimation. This allows us to write strictly convex, quadratic inequalities in the form $||Rx||_2^2 - 2a^T x \leq \alpha$. By solving $R^T R x_0 = a$, we find bounds $\|R(x - x_0)\|_2 \leq \delta := \sqrt{\max\{0, \alpha + a^T x_0\}}$, from which one can find the componentwise bounds $|x - x_0| \leq \delta \operatorname{diag}(R^{-1} R^{-T})$.

4 Open conjectures

We end by listing some open conjectures which are likely to be tractable in the near future by computer-assisted techniques.

The Wright conjecture (1955).
For $0 < p < \pi/2$, every solution of the delay-differential equation $x'(t) = -p(1 + x(t))x(t - 1)$ tends to zero as $t \to \infty$.

This conjecture by WRIGHT [17] is currently under attack in Szeged, and appears to be "almost" (i.e., up to final correctness checks) solved.

Hilbert's 16th Problem, second part (1900).
"This is the question as to the maximum number and position of Poincaré's boundary cycles (cycles limites) for a differential equation of the first order and degree of the form

$$\frac{dy}{dx} = \frac{Y}{X},$$

where X and Y are rational integral functions of the n-th degree in x and y. Written homogeneously, this is

$$X(y\frac{dz}{dt} - z\frac{dy}{dt}) + Y(z\frac{dx}{dt} - x\frac{dz}{dt}) + Z(x\frac{dy}{dt} - y\frac{dx}{dt}) = 0,$$

where X, Y, and Z are rational integral homogeneous functions of the n-th degree in x, y, z, and the latter are to be determined as functions of the parameter t." (quoted from HILBERT [7]). The problem has been reposed in 2000 by SMALE [16] as one of the "Mathematical Problems for the Next Century". For a recent survey, see ILYASHENKO [8].

It is known that the number of limit cycles is finite in each particular instance, but no finite upper bound is known. This problem is still unsolved, even for $n = 2$, in spite of a lot of research on the problem. For $n = 2$, the largest number of limit cycles known is 4, and probably this bound is tight. For $n = 2$, the problem amounts to understanding the set of periodic solutions of pairs of autonomous differential equations

$$\dot{x} = ax^2 + bxy + cy^2 + dx + ey + f,$$

$$\dot{y} = a'x^2 + b'xy + c'y^2 + d'x + e'y + f'.$$

This is a 12-dimensional manifold of orbits, which can be reduced to 5 dimensions by suitable linear transformation of the variables. Periodic solutions can be described as fixed points of Poincaré maps. Hence the problem is to show that certain equations $F(z, w) = z$ in the single variable z and parameterized by a 5-dimensional vector w never have more than 4 solutions (except when they have infinitely many – a degenerate, well understood case excloded by the demand that the periodic solution is a limit cycle).

The low dimension makes the problem appear feasible for branch and bound techniques. Some obstacles remain: The fixed point equation is not algebraic but a Poincaré mapping must be enclosed, and the parameter vector ranges over an unbounded 5-dimensional manifold, whence additional asymptotic estimates are needed.

The Thompson-Smith problem (2000).
Is 10 or 12 the minimim norm of the Thompson-Smith lattice in dimension 248?

The conjectured answer 12 (for background information see LEMPKEN et al. [9]) would follow (according to Gabriele Nebe, personal communication) from the solvability of a nonlinear mixed integer constraint satisfaction problem. The integer coefficients are generated automatically by Maple; some have several hundred digits.

5 Conclusion

As we have seen, there is lots of interesting work to be done at the interface between

- **Mathematics** – the art and science of unambiguous concepts,
- **Logic** – the art and science of impeccable reasoning,
- **Operations Research** – the art and science of optimal planning,
- **Numerical Analysis** – the art and science of algorithmic approximation, and
- **Computer Science** – the art and science of efficient computation.

References

[1] O. Caprotti and M. Oostdijk, Formal and Efficient Primality Proofs by Use of Computer Algebra Oracles, J. Symbolic Computation 32 (2001), 55-70.

[2] F. Domes and A. Neumaier, GLOPTLAB – a Matlab-based global optimization laboratory. In preparation (2007).

[3] M.C. Ferris, G. Pataki and S. Schmieta, Solving the Seymour problem, Optima 6 (2001), 2.

[4] A. Frommer, Proving conjectures by use of interval arithmetic, pp. 1–13 in: Perspective on Enclosure Methods (U. Kulisch et al., eds.), Springer, Wien 2001.

[5] T.C. Hales, A Proof of the Kepler Conjecture, Annals of Mathematics 162 (2005), 1065-1185.
Computations are described in the special issue: Discrete and Computational Geometry, 36 (2006), 1-265.

[6] J. Hass, M. Hutchings, and R. Schlafly, Double bubbles minimize, Ann. Math. (2) 515 (2000), 459–515.
`http://math.ucdavis.edu/~hass/bubbles.html`

[7] D. Hilbert, Mathematical Problems, Bull. Amer. Math. Soc. 8 (1901/2), 437–479.

[8] Yu. Ilyashenko, Centennial history of Hilbert's 16th problem, Bull. Amer. Math. Soc. 39 (2002), 301-354.

[9] W. Lempken, B. Schröder and P.H. Tiep, Symmetric Squares, Spherical Designs, and Lattice Minima, J. Algebra, 2001

[10] A. Neumaier, Interval methods for systems of equations, Cambridge Univ. Press 1990.

[11] S.M. Rump, INTLAB – INTerval LABoratory, pp. 77–104 in: Developments in reliable computing (T. Csendes, ed.), Kluwer, Dordrecht 1999.
`http://www.ti3.tu-harburg.de/rump/intlab/index.html`

[12] S.M. Rump, Verification methods for dense and sparse systems of equations, pp. 63-136 in: J. Herzberger (ed.), Topics in Validated Computations – Studies in Computational Mathematics, Elsevier, Amsterdam 1994.

[13] H. Schichl, The COCONUT environment. http://www.mat.univie.ac.at/coconut-environment/

[14] H. Schichl and A. Neumaier, Transposition theorems and qualification-free optimality conditions, SIAM J. Optimization, to appear (2006). http://www.mat.univie.ac.at/~neum/papers.html#trans

[15] SEDUMI web site. http://sedumi.mcmaster.ca/

[16] S. Smale, Mathematical Problems for the Next Century, Math. Intelligencer 20, No. 2, 7-15, 1998.

[17] E. M. Wright, A nonlinear difference-differential equation, J. Reine Angew. Math. 194 (1955), 66-87.

On the solution to numerical problems using stochastic arithmetic

René Alt, Jean-Luc Lamotte
CNRS, UMR 7606, LIP6
University Pierre et Marie Curie
4 place Jussieu
75252 Paris cedex 05, France
Rene.Alt@lip6.fr
Jean-Luc.Lamotte@lip6.fr

Svetoslav Markov
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
"G. Bonchev" st.
bl. 8, 1113 Sofia, Bulgaria
smarkov@bio.bas.bg

Abstract

We investigate some algebraic properties of the system of stochastic numbers with the arithmetic operations addition and multiplication by scalars and the relation inclusion and point out certain practically important consequences from these properties. Our idea is to start from a minimal set of empirically known properties and to study these properties by an axiomatic approach. Based on this approach we develop an algebraic theory of stochastic numbers. A numerical example based on the Lagrange polynomial demonstrates the consistency between the CESTAC method and the presented theory of stochastic numbers.

1. Introduction

The CESTAC method is a widely used statistical Monte-Carlo type method to compute the accuracy of the solution of real life numerical problems implemented on a computer. In this method an imprecise numbers is represented as an N-tuple of random values with Gaussian distribution so that the mean value and standard deviation of this N-tuple provide respectively approximations of the exact unknown value and of the error. Such an N-tuple which is a sampling of a Gaussian random variable is named in this context a *discrete stochastic number*. In practice the CESTAC method has been implemented in a software called CADNA in which the N samples of the stochastic numbers are randomly rounded up or down so as to take into account the round-off errors, see [3], [8], [9], [12], with the same idea that directed rounding is used for implementing interval arithmetic [5].

In order to provide a good algebraic understanding of the performance of the CESTAC method discrete stochastic numbers have been modeled as Gaussian random variables with a known mean value and a known standard deviation named here *stochastic numbers*. Thus stochastic numbers are idealizations of discrete stochastic numbers. To get an idea of the operations between stochastic numbers it should be mentioned that we are concerned with imprecise data in which the unknown errors are relatively important (say, of order 10^{-2}–10^{-3}) whereas the arithmetic operations are performed using double precision arithmetic. In this case it can be easily checked that addition and multiplication by reals in the CESTAC method satisfy certain algebraic properties, e. g. stochastic numbers form a commutative and cancellative monoid with respect to addition. Some fundamental properties of stochastic numbers are considered in [4], [10].

This work is part of a more general one, which aims,to study the abstract algebraic structures induced by the operations on stochastic numbers and to compare the relevant theory with the performance of the CESTAC method [1], [2], [6], [7]. In the paper we restrict ourselves to the arithmetic operations addition, negation and multiplication by scalars and the relation inclusion.

1.1. Stochastic Arithmetic (*CSA*): basic properties

Denote by S the set of stochastic numbers and by S^n the set of all n-tuples of stochastic numbers. The operations addition and multiplication by -1 (negation) are well defined in S, resp. S^n. We next recall some basic properties of these operations and derive some logical consequences of these properties, which will allow us to better understand the nature of stochastic numbers. The system $(S^n, +)$ is a commutative monoid (semigroup with null) with cancellation law. The operator negation is an automorphism $\neg : S^n \to S^n$, that is: $\neg(A + B) = \neg A + (\neg B)$, and involution: $\neg(\neg A) = A$. These properties can be checked experimentally, say, by a CESTAC-like method, see e. g.

[8], [9], [10]. So, we next consider these properties as axiomatically given, and we want to derive some simple consequences.

We first note that S^n is not a group with respect to addition; however it can be easily embedded in a group. The standard algebraic construction that converts any abelian monoid with cancellation law into a group will be further referred as embedding construction. Recall that this approach is used to pass from the monoid of nonnegative reals $(\mathbb{R}^+, +)$ to the set of reals $(\mathbb{R}, +)$. Thus, it is natural instead of the original system $(S^n, +)$ to consider the extended system $(\mathbb{S}^n, +)$ obtained by the embedding construction. We next briefly recall this construction.

1.2. Algebraic completion of the monoid of stochastic numbers

Every abelian monoid $(M, +)$ with cancellation law induces an abelian group $(\mathbb{M}, +)$, where $\mathbb{M} = M^2/\sim$ is the *difference (quotient) set* of M consisting of all pairs (A, B) factorized by the congruence relation $\sim$: $(A, B) \sim (C, D)$ iff $A+D = B+C$, for $A, B, C, D \in M$. Addition in $\mathbb{M}$ is defined by $(A, B) + (C, D) = (A + C, B + D)$. The neutral (null) element of $\mathbb{M}$ is the class (Z, Z), $Z \in M$. Due to the existence of null element in M, we have $(Z, Z) \sim (0, 0)$. The opposite element to $(A, B) \in \mathbb{M}$ is $\text{opp}(A, B) = (B, A)$. The mapping $\varphi : M \longrightarrow \mathbb{M}$ defined for $A \in M$ by $\varphi(A) = (A, 0) \in \mathbb{M}$ is an *embedding* of monoids. We *embed* M in $\mathbb{M}$ by identifying $A \in M$ with the equivalence class $(A, 0) \sim (A + X, X)$, $X \in M$; all elements of $\mathbb{M}$ admitting the form $(A, 0)$ are called *proper* and the remaining (new) elements are called *improper*. The set of all proper elements of $\mathbb{M}$ is $\varphi(M) = \{(A, 0) \mid A \in M\} \cong M$.

Using the embedding construction described above the system $(S^n, +)$ is embedded into a group $(\mathbb{S}^n, +)$ in a unique way. We define multiplication by -1, called *negation*, in the group $\mathbb{S}^n$, by means of: $\neg(A, B) = (\neg A, \neg B)$, $A, B \in S^n$. In what follows we shall use lower case roman letters to denote the elements of $\mathbb{S}^n$, writing e. g. $a = (A_1, A_2), A_1, A_2 \in \mathbb{S}^n$. Thus negation will be denoted as $\neg a$; $a \neg b$ means $a + (\neg b)$. Clearly the properties of negation in S hold also in $\mathbb{S}$, that is: $\neg(a + b) = \neg a \neg b$ and $\neg(\neg a) = a$. From $\text{opp}(a) + a = 0$ we obtain $\neg\text{opp}(a) \neg a = 0$, that is $\neg\text{opp}(a) = \text{opp}(\neg a)$. The element $\neg\text{opp}(a) = \text{opp}(\neg a)$ is further denoted by a_- and the corresponding operator is called *dualization* or *conjugation*. We say that a_- is the conjugate (or dual) of a. In the sequel we shall express the opposite element symbolically as: $\text{opp}(a) = \neg a_-$, minding that $a + (\neg a_-) = 0$ (to be briefly written $a \neg a_- = 0$).

In Section 2 we investigate the system $(\mathbb{S}^n, +, \neg)$ obtained by algebraic completion by means of a novel approach. Namely, starting from a minimal set of basic algebraic properties, we naturally arrive to the necessity of studying separately the spaces of mean values (which is a vector space) and of standard deviations (which is an s-space). In Section 3 we consider the system $(\mathbb{S}^n, +, \mathbb{R}, *)$. There we also discuss an algebraically natural approach to define an order relation inclusion for stochastic numbers arriving thus to the system $(\mathbb{S}^n, +, \mathbb{R}, *, \subseteq)$. In Section 4 we give some numerical examples aiming to compare the CSA theory with the performance of the CESTAC method. Our numerical experiments with Lagrange interpolation demonstrate that the model of stochastic numbers is, at least in the case of the operationsin consideration, in perfect agreement with the results obtained with the CESTAC method.

2. Decomposing the group of stochastic numbers

We shall now concentrate on the algebraic properties of the system $(\mathbb{S}^n, +, \neg)$. For a better understanding the following reminder about expressions involving the dual operator will be useful: i) $a + \text{opp}(a) = 0$ is equivalent to $a_- \neg a = 0$ or $a \neg a_- = 0$; ii) $(a + b)_- = a_- + b_-$, iii) $y \neg y = 0$ is equivalent to $y = y_-$; iv) $\neg z = z$ is equivalent to $z + z_- = 0$. An element y with property iii) is called *linear* or *distributive*; an element z with property iv) is called *centred* or *0-symmetric*.

Denote $nx = x + x + ... + x$ (n times). We recall that a divisible (additive) group is such that every equation of the form $nx = a$ has a solution x for any $a \in G$; the solution will be further denoted $(1/n)a$. An (additive) group is torsion-free if $na = 0$ implies $a = 0$ for any $a \in G$.

Remark. Clearly, the group of stochastic numbers is divisible and torsion-free (the monoid possesses the same properties). For our purposes it will be sufficient that the following two properties hold in $(G, +)$: i) $x + x = a \Longrightarrow x = (1/2)a$, and ii) $x + x = 0 \Longrightarrow x = 0$. Note that the latter property is equivalent to $x + x = y + y \Longrightarrow x = y$ (indeed, $x + x = y + y \Longrightarrow (x \neg y_-) + (x \neg y_-) = 0 \Longrightarrow (x \neg y_-) = 0 \Longrightarrow x = y$).

Let $(G, +)$ be an additive abelian divisible torsion-free group. In addition we shall assume that G posseses an involutary automorphism $\neg : G \to G$, such that: C1. $\neg(a + b) = \neg a \neg b$; C2. $\neg(\neg a) = a$.

Remark. In particular, the operator "$\neg$" may coincide with opposite or identity. Conditions C1–C2 imply $\neg 0 = 0$ (to see this substitute $b = \neg a_-$ in C1).

Theorem (Decomposition theorem). G is an additive divisible torsion-free abelian group with an involutary automorphism "$\neg$". For every $x \in G$ there exist unique $y, z \in G$, such that: i) $x = y + z$; ii) $y \neg y = 0$; iii) $\neg z = z$; iv) $y = z \Longrightarrow y = z = 0$.

Proof. Let us consider equations i)–iii) as a system of equations for x, y, z and let us solve this system with respect to y and z. To this end we shall repeatedly use some properties of the dual operator, such that $\mathrm{opp}(a) = \neg a_-$, equation $y \neg y = 0$ is equivalent to $y = y_-$ and $\neg z = z$ is equivalent to $z + z_- = 0$.

Write $x = y + z$ in the form: $x_- = y_- + z_-$; replacing y_- by y and z by $\neg z$ we obtain: $x_- = y \neg z_-$. Adding the latter equation to $x = y + z$ we obtain

$$x + x_- = y + y. \tag{1}$$

Similarly, write $x = y + z$ in the form :$\neg x = \neg y + \neg z$; replacing y_- by y and z by $\neg z$ we obtain: $\neg x = \neg y_- + z$. Adding the latter equation to $x = y + z$ we obtain

$$x \neg x = z + z. \tag{2}$$

Summing up equations (1) and (2) we obtain $x + x = y + y + z + z$, which implies $x = y + z$ (using that $x + x = 0 \Longrightarrow x = 0$).

Assume now that a $x \in G$ is given. Chose y and z to satisfy resp. (1) and (2). Such elements exist due to the divisibility assumption; denote them $y = (1/2)(x + x_-)$ and $z = (1/2)(x \neg x)$. Clearly, y is linear, whereas z is centred. As we have $x = y + z$, it follows that any $x \in G$ is decomposable into a sum of a linear and a centred element. To show that the sum is direct, it remains to prove that $y = z \Longrightarrow y = z = 0$. Assume $y = z$. Then $y + y = z + z$, or $x + x_- = x \neg x$. The latter implies $x_- = \neg x$, or $x = \neg x_- = \mathrm{opp}(x)$, or $x + x = 0$, or $x = 0$ (again due to the assumption $x + x = 0 \Longrightarrow x = 0$). □

Comments. The Decomposition theorem states that any $x \in G$ can be written in the form:

$$x = y + z = (1/2)(x + x_-) + (1/2)(x \neg x). \tag{3}$$

The element $y = (1/2)(x + x_-)$ satisfies $y = y_-$, equivalently $y \neg y = 0$; thus y is a linear (distributive) element.

Alternatively, the element $z = (1/2)(x \neg x)$ satisfies $z = \neg z$, equivalently $z + z_- = 0$; therefore z is centred (0-symmetric).

The subset of all linear elements of G is denoted $G' = \{y \in G \mid y \neg y = 0\}$ and the subset of all centred elements of G is denoted $G'' = \{z \in G \mid z = \neg z\}$.

If negation coincides with opposite in G, then $x \neg x = 0$ for all $x \in G$ and, from (3): $x = (1/2)(x + x_-)$ for all $x \in G$. In this case the subset G'' is empty and G consists only of linear elements. Alternatively, if negation coincides with identity in G, then $x + x_- = 0$ for all $x \in G$ and, from (3): $x = (1/2)(x \neg x)$ for all $x \in G$. In this case the subset G' is empty and G consists only of centred elements.

Corollary. Let G be an additive divisible torsion-free abelian group with an involutary automorphism "¬". Then G is a direct sum of $G' = \{y \in G \mid y \neg y = 0\}$ and $G'' = \{z \in G \mid z = \neg z\}$, symbolically $G = G' \bigoplus G''$.

Remark. Note that while the group $G = (G, +, \neg)$ possesses an operator (negation) in addition to opposite, the subgroups G', G'' do not possess additional operator (as negation coincides with opposite in G' and with identity in G''). Therefore we do not need to write down the operator negation in the groups: $G = G' \bigoplus G''$ can be written as $(G, +, \neg) = (G', +) \bigoplus (G'', +)$.

2.1. Practical consequences

The Decomposition theorem implies that the algebraically natural presentation of stochastic numbers is as a sum of two components — a linear component and a centred component. The linear component, by definition, is such that negation of this element coincides with opposite, and the centred component is such that negation coincides with identity. The latter holds for elements of $\mathbb{S}$, but it also holds for elements of S (that is proper stochastic numbers). Indeed, by the embedding construction, proper stochastic numbers as elements of $\mathbb{S}$ are pairs of the form $(A, 0)$, wherein $A \in S$. Assume first that $(A, 0)$ is linear, that is $(A, 0) \neg (A, 0) = 0$; this implies $A \neg A = 0$, that is negation of A coincides with opposite. Such property have, e. g., real numbers (real vectors) as elements of a linear space. Assume now that $(A, 0)$ is centred, that is $(A, 0) = \neg (A, 0)$; this means $A = \neg A$, that is negation of A coincides with identity.

According to the Corollary the group of stochastic numbers $\mathbb{S}$ decomposes as a direct sum of two subgroups $\mathbb{S} = \mathbb{S}' \bigoplus \mathbb{S}''$. Thus a stochastic number $a \in \mathbb{S}$ can be written in the form $a = (a'; a'')$. As well-known, the first component a' is interpreted as mean value, and the second component a'' is the standard deviation. A stochastic number of the form $(a'; 0)$ has zero standard deviation and represents a (pure) mean value, whereas a stochastic number of the form $(0; a'')$ has zero mean value and represents a (pure) standard deviation. Using addition in $\mathbb{S}$, as defined in a direct sum by means of $(a'; a'') + (b'; b'') = (a' + b'; a'' + b'')$, we have the presentation $a = (a'; a'') = (a'; 0) + (0; a'')$. For negation we have $\neg a = \neg(a'; a'') = (\neg a'; \neg a'') = (-a'; a'')$, minding that $\neg a' = -a'$ and $\neg a'' = a''$.

In the case of stochastic vectors we have: $a = (a'; a'') \in \mathbb{S}^n$ with $a' \in \mathbb{R}^n$, $a'' \in \mathbb{R}^n$. We have $a'' \geq 0$ for a proper; a'' has at least one negative component for improper a.

The two systems $(\mathbb{S}', +)$, $(\mathbb{S}'', +)$ composing $(\mathbb{S}, +, \neg)$ are of distinct algebraic nature, hence it is correct to use different notations for the operation addition. The system of mean values is identified with the additive group of reals $(\mathbb{R}, +)$ where negation coincides with opposite: $\neg\delta = -\delta$ for $\delta \in \mathbb{R}$. Therefore we shall use the usual sign for addition in the system $(\mathbb{S}', +)$, but we use a distinct sign for addition

in the system $(\mathbb{S}'', +)$, namely $(\mathbb{S}'', \oplus)$.

In order to characterize the system $(\mathbb{S}'', \oplus)$ define for $\alpha \in \mathbb{R}$ the *sign function* σ by: $\sigma(\alpha) = \{+, \text{ if } \alpha \geq 0; \ -, \text{ if } \alpha < 0\}$. As shown in [6], [7], the system $(\mathbb{S}'', \oplus)$ is identified with the group system $(\mathbb{R}, \oplus)$, where for $\alpha, \beta \in \mathbb{R}$ addition is defined by:

$$\alpha \oplus \beta = \sigma(\alpha + \beta)\sqrt{|\sigma(\alpha)\alpha^2 + \sigma(\beta)\beta^2|}, \tag{4}$$

and negation is identity: $\neg\delta = \delta$ for $\delta \in \mathbb{R}$.

Define the *symmetric square and square root functions* by: $\alpha^{\widehat{2}} = \sigma(\alpha)\alpha^2$, $\alpha^{\widehat{1/2}} = \sigma(\alpha)|\alpha|^{1/2}$, $\alpha \in \mathbb{R}$. If $\alpha \geq 0$ then $\alpha^{\widehat{2}}$ coincides with α^2, similarly $\alpha^{\widehat{1/2}}$ and $\alpha^{1/2}$ are identical; however if $\alpha < 0$ then $\alpha^{\widehat{2}} = -\alpha^2$, $\alpha^{\widehat{1/2}} = -\alpha^{1/2}$. Using the symmetric square and square root functions (4) can be written as

$$\alpha \oplus \beta = (\alpha^{\widehat{2}} + \beta^{\widehat{2}})^{\widehat{1/2}}. \tag{5}$$

For two stochastic numbers $(m_1; s_1), \ (m_2; s_2) \in \mathbb{S}$, we have

$$(m_1; s_1) + (m_2; s_2) = (m_1 + m_2; (s_1^{\widehat{2}} + s_2^{\widehat{2}})^{\widehat{1/2}}), \tag{6}$$

$$\neg(m_1; s_1) = (-m_1; \ s_1). \tag{7}$$

For the special case of two proper stochastic numbers $(m_1; s_1), \ (m_2; s_2), s_1, s_2 \geq 0$, formula (6) becomes

$$(m_1; s_1) + (m_2; s_2) = (m_1 + m_2; \ \sqrt{s_1^2 + s_2^2}). \tag{8}$$

This is the familiar formula for the addition of two independent random variables with normal distribution. The advantage of formula (6) is that it can be applied in the general case without assuming nonnegativity of standard deviations.

3. The space $(\mathbb{S}^n, +, \mathbb{R}, *, \subseteq)$

The Decomposition theorem states that a group with negation is a direct sum of two spaces under mild assumptions for the group (divisibility, torsion-freedom) and the negation operator (automorphism, involution). As the groups $\mathbb{S}$, resp. $\mathbb{S}^n$, satisfy these assumptions (as known from empirical evidence), we can state that $\mathbb{S} = \mathbb{S}' \bigoplus \mathbb{S}''$, resp. $\mathbb{S}^n = (\mathbb{S}')^n \bigoplus (\mathbb{S}'')^n$, where $\mathbb{S}'$ is the space of mean values and $\mathbb{S}''$ is the space of standard deviations.

3.1. Multiplication by scalars

Multiplication by scalars "$*$" is defined for $x = (m; s) \in \mathbb{S}$ by

$$\gamma * x = \gamma * (m; s) \stackrel{def}{=} (\gamma m; \ |\gamma|s), \ \gamma \in \mathbb{R}. \tag{9}$$

Clearly, the system $(\mathbb{S}', +, \mathbb{R}, *)$ is a linear space and will be written also as $(\mathbb{S}', +, \mathbb{R}, \cdot)$. As we know the system $(\mathbb{S}'', \oplus, \mathbb{R}, *)$, resp. $((\mathbb{S}'')^n, \oplus, \mathbb{R}, *)$, with operation addition defined by (4) and multiplication by scalars defined from (9) by

$$\gamma * s = |\gamma|s, \ \gamma \in \mathbb{R}, \tag{10}$$

is an *s-space* in the sense of the following definition [6]:

Definition. The system $(\mathcal{G}, \oplus, \mathbb{R}_D, *)$ is called an s-space if $(\mathcal{G}, \oplus)$ is an abelian group, such that for $s, t \in \mathcal{G}$, $\alpha, \beta \in \mathbb{R}$:

$$\alpha * (s \oplus t) = \alpha * s \oplus \alpha * t, \tag{11}$$

$$\alpha * (\beta * s) = (\alpha\beta) * s, \tag{12}$$

$$1 * s = s, \tag{13}$$

$$(-1) * s = s, \tag{14}$$

$$\sqrt{\alpha^2 + \beta^2} * s = \alpha * s \oplus \beta * s. \tag{15}$$

Let us define in $(\mathbb{S}'', \oplus, \mathbb{R}, *)$ two operations as follows:

$$\alpha + \beta = (\alpha^{\widehat{1/2}} \oplus \beta^{\widehat{1/2}})^{\widehat{2}}, \tag{16}$$

$$\gamma \cdot s = \gamma * s_{\sigma(\gamma)}, \ \gamma \in \mathbb{R}. \tag{17}$$

It can be easily checked that the space $(\mathbb{S}'', +, \mathbb{R}, \cdot)$ with operations "$+$" and "$\cdot$" defined by (16), (17) is linear. Thus the space $(\mathbb{S}'', \oplus, \mathbb{R}, *)$ can be considered as a linear space with two additional operations, namely $(\mathbb{S}'', +, \oplus, \mathbb{R}, \cdot, *)$.

3.2. Inclusion

We next discuss two relations for inclusion of stochastic numbers. The so-called *interval inclusion* (briefly: i-inclusion) is defined for $x_1 = (m_1; s_1), \ x_2 = (m_2; s_2) \in \mathbb{S}$, by:

$$x_1 \subseteq_i x_2 \Longleftrightarrow |m_2 - m_1| \leq s_2 - s_1. \tag{18}$$

We note that addition is inverse i-inclusion isotone, that is: $x_1 + y \subseteq_i x_2 + y$ implies $x_1 \subseteq_i x_2$ [2]. However, it is easy to see that i-inclusion isotonicity does not hold, i. e. $x_1 \subseteq_i x_2$ does not imply $x_1 + y \subseteq_i x_2 + y$. If we want that a two-directional implication

$$x_1 \subseteq x_2 \Longleftrightarrow x_1 + y \subseteq x_2 + y$$

holds in $\mathbb{S}$, then instead of "$\subseteq_i$" we should use the inclusion relation "$\subseteq_s$" between two stochastic numbers defined by

$$x_1 \subseteq_s x_2 \Longleftrightarrow |m_2 - m_1|^2 \leq s_2{}^{\widehat{2}} - s_1{}^{\widehat{2}}. \tag{19}$$

Relation (19) is called *stochastic inclusion*, briefly: s-inclusion. In the proper case s-inclusion has been proposed in [1].

Proposition 1. Addition and multiplication by scalars are (inverse) inclusion isotone (invariant with respect to s-inclusion).

The proof in the general case (s real) is similar to the one for the proper case (s nonnegative) see [1].

Remark. In fact it can be seen that $x_1 \subseteq_s x_2$ if $x_2 = x_1 + y$ for some stochastic number $y = (m, s)$ whose "range" $[m - s, m + s]$ contains 0.

We shall next compare relations (19) and (18). To this end we introduce an end-point presentation.

End-point presentation. This presentation may be useful when dealing with confidence intervals. The confidence interval corresponding to the stochastic number $(m; s)$ is $[m - \gamma s, m + \gamma s]$, where $\gamma > 0$ is a chosen number (usually $\gamma \approx 2$). For simplicity in the sequel we assume $\gamma = 1$, which corresponds to usual compact intervals on $\mathbb{R}$.

Recall that the relation between the end-point presentation of an interval $A = [a^-, a^+] \subseteq \mathbb{R}$ and its mid-point/radius presentation $A = (a';\ a'')$ is given by:

$$\begin{aligned} a^- &= a' - a'', \qquad a^+ = a' + a''; \\ a' &= (a^- + a^+)/2, \ \ a'' = (a^+ - a^-)/2. \end{aligned}$$

Recall also the relation $a^+a^- = a'^2 - a''^2$.

The i-inclusion (18) admits a simple end-point presentation, namely for $A \subseteq_i B$ condition $|b' - a'| \leq b'' - a''$ is presented in end-point form as $b^- \leq a^-$, $a^+ \leq b^+$. We next look for an end-point presentation for the s-inclusion (19): $A \subseteq_s B \Longleftrightarrow (b' - a')^2 \leq b''^2 - a''^2$.

The condition $(b' - a')^2 \leq b''^2 - a''^2$ can be written as $b'^2 - b''^2 + a'^2 + a''^2 \leq 2a'b'$. Replacing $b'^2 - b''^2 = b^+b^-$, $a' = (a^- + a^+)/2$, $a'' = (a^+ - a^-)/2$, etc. we obtain: $2b^+b^- + {a^+}^2 + {a^-}^2 \leq (a^+ + a^-)(b^+ + b^-)$. Thus the end-point condition for s-inclusion obtains the form:

$$A \subseteq_s B \Longleftrightarrow {a^+}^2 + {a^-}^2 + 2b^+b^- \leq (a^+ + a^-)(b^+ + b^-),$$

which can be also written in the form $2(b^+b^- - a^+a^-) \leq (a^+ + a^-)(b^+ + b^- - (a^+ + a^-))$.

Proposition 2. Interval inclusion (18) implies stochastic inclusion (19).

Proof. We sketch the proof for proper stochastic numbers. Assume that $A = (a';\ a'')$ is i-included in $B = (b';\ b'')$, $A \subseteq_i B$, which according to (18) means $|b' - a'| \leq b'' - a''$. We have to show that (19) holds true. Note first that from (18) we have $0 \leq a'' \leq b''$. Now from $|b' - a'| \leq b'' - a''$ we have $(b' - a')^2 \leq (b'' - a'')^2 \leq (b'' - a'')(b'' + a'') = b''^2 - a''^2$. □

As a consequence from Proposition 2, stochastic addition is i-inclusion isotone.

3.3. Lattice operations

The lattice operations for the i-inclusion are well-known. We next consider the lattice operations for the s-inclusion, sketching the results for proper intervals. The case when one of the stochastic number is s-included in the other one is obvious. Let us determine $\sup(A, B) = C$ for the case when neither $A \subseteq_s B$, nor $B \subseteq_s A$.

Recall that in the case of i-inclusion we have :

$$c'' = |c' - a'| + a'' = |c' - b'| + b''.$$

From $|c' - a'| + a'' = |c' - b'| + b''$ we can compute first c' and then $c'' = |c' - a'| + a''$.

Similarly in the case of s-inclusion we have:

$$c''^2 = a''^2 + (c' - a')^2 = b''^2 + (b' - c')^2.$$

From $a''^2 + (c' - a')^2 = b''^2 + (b' - c')^2$ we compute c':

$$2c' = \frac{b'^2 + b''^2 - (a'^2 + a''^2)}{b' - a'} = \frac{b''^2 - a''^2}{b' - a'} + b' + a'.$$

For $c''^2 = a''^2 + (c' - a')^2$ we obtain:

$$4c''^2 = \frac{(b''^2 - a''^2)^2}{(b' - a')^2} + 2(b''^2 + a''^2) + (b' - a')^2.$$

Let us now determine $\inf(A, B) = D$ for the case when neither $A \subseteq_s B$, nor $B \subseteq_s A$.

Recall first that in the case of i-inclusion we have:

$$d'' = a'' - |d' - a'| = b'' - |d' - b'|.$$

From $a'' - |d' - a'| = b'' - |d' - b'|$ we can compute first d' and then $d'' = a'' - |d' - a'|$.

Similarly in the case of s-inclusion we have:

$$d''^2 = a''^2 - (d' - a')^2 = b''^2 - (b' - d')^2.$$

From $a''^2 - (d' - a')^2 = b''^2 - (b' - d')^2$ we compute d':

$$2d' = \frac{b'^2 - b''^2 - (a'^2 - a''^2)}{b' - a'} = -\frac{b''^2 - a''^2}{b' - a'} + b' + a'.$$

For $d''^2 = a''^2 - (d' - a')^2$ we obtain:

$$4d''^2 = -\frac{(b''^2 - a''^2)^2}{(b' - a')^2} + 2(b''^2 + a''^2) - (b' - a')^2.$$

Clearly, we have $d'' < 0$ for relatively small standard deviations a'', b'' and relatively large value of $|b' - a'|$. This means that $\inf(A, B)$ can have a negative standard deviation for proper stochastic numbers. In other words two proper stochastic numbers may not have a proper infimum.

4. Application: Lagrange interpolation

The goal of this section is to compare the results obtained with the theory developed in this paper, which is named *continuous stochastic arithmetic (CSA)*, with respective results obtained with the CESTAC method and with interval arithmetic.

As mentioned in the introduction, in the CESTAC method, each stochastic variable is represented by a N-tuple of gaussian random values with known mean value m and standard deviation s. The method also uses a special arithmetic called *discrete stochastic arithmetic (DSA)*, which acts on the N-tuples.

In the scope of granular computing [11], stochastic arithmetic CSA operates on stochastic numbers and is directly derived from operations on independent gaussian random variables. Hence a stochastic number is a granule and CSA is a tool for computing with granules.

With the same point of view of granular computing, in *discrete stochastic arithmetic (DSA)* a granules is composed by a N-tuple of samples of the same mathematical result of an arithmetical operator implemented in floating point arithmetic. The samples differ from each other because data are imprecise and because of different rounding. The operator acting on these granules is a floating point operator corresponding to the exact arithmetical operator which is performed N times in a synchronous way with random rounding. Thus the result is also a granule named discrete stochastic number. It has been shown that *DSA* operating on discrete stochastic numbers has many properties (but not all) of real numbers; In particular the notion of stochastic zero has been defined. As explained above, the CADNA library merely implements *DSA*.

To compare the two models, a specific library has been developed which implements both continuous and discrete stochastic arithmetic. The computations are done separately. *CSA* implements the mathematical rules defined above in sections 2–3.

The comparison has been done on the Lagrangian interpolation method. Let $(x_i,\ y_i)$, $i = 0, 1, ..., n$, be a set of $n + 1$ pairs of numbers where all x_i are different. We want to compute the value $p(t)$ of the Lagrangian polynomial at a given point t with the classical formula:

$$p(t) = y_0 l_0(t) + y_1 l_1(t) + \cdots + y_n l_n(t), \tag{20}$$

wherein

$$l_i(t) = \frac{\prod_{i \neq j}(t - x_j)}{\prod_{i \neq j}(x_i - x_j)}.$$

We consider the situation when the values of y_i are imprecise (contain some errors) and x_i are considered exact. This case is within the scope of our theoretical results where only addition or subtraction between two stochastic numbers and multiplication of a stochastic number by a real number is considered.

For all examples presented below, we take $n = 10$; the exact x-values are defined as $x_i = i,\ i = 1, ..., n + 1$, and the imprecise values y_i are around 1. The latter means that in the interval case all intervals y_i have a midpoint 1, whereas in the stochastic case they have a mean value 1.

4.1 Interval approach

First, the interval approach is considered to obtain exact bounds for the results assuming that some guaranteed bounds are given for the data y_i in the form of intervals Y_i, that is $y_i \in Y_i$.

Then it is well-known that at each t

$$p(t) \in P(t) = l_0(t) * Y_0 + l_1(t) * Y_1 + \cdots + l_n(t) * Y_n.$$

The computation of the interval polynomial $P(t)$ has been performed with the Intlab implementation [5] of interval arithmetic. The maximum error on the Y_i value is $ierr = 0.02$. With the case $Y_i = [1 - ierr; 1 + ierr] =$ constant and $x_i = i,\ i = 1, ..., 11$, the upper and lower bounds of P are shown in black on Fig. 1.

Remark. This is an example when the use of so-called naive interval arithmetic produces exact (sharp) bounds (Fig. 1). Normally, naive interval arithmetic produces pessimistic bounds. In most cases, such sharp bounds cannot be obtained by naive interval arithmetic and more sophisticate methods should be used.

4.2 Experiments with stochastic arithmetic

The computation of (20) is done using the stochastic arithmetic approach (CSA). This approach is based on the gaussian random variable (m, σ) with a mean value m and a standard deviation σ. It is well known that 95% of the samples of a such variable are inside the interval $[m - 2\sigma, m + 2\sigma]$. To compare the results with the interval approach, the Y_i are equal to $(1, 0.01)$.

The computation has been performed with our specific implementation of CSA. A set of $(m_{p(t_i)}, \sigma_{p(t_i)})$ is obtained. The gray lower and upper curves in the figure 1 represent the results of *CSA*, i.e. each point of the lower curve (resp. the upper curve) is equal to $m_{p(t_i)} - 2 * \sigma_{p(t_i)}$ (resp. $m_{p(t_i)} + 2 * \sigma_{p(t_i)}$).

It can be seen that the two curves corresponding to *CSA* are inside the range of those computed with Intlab.

4.3 Discrete stochastic arithmetic

The results computed with *DSA* are compared with *CSA* in figures 2, 3, 4, 5. On each figure, the N samples and the lower and upper curve obtained with *CSA* are drawn respectively for $N = 3, 5, 10, 20$. All the figures are composed of two sub-figures. The left sub-figure shows the N curves of the samples. The right sub-figure compares the computed mean value and standard deviation obtained from the N-samples and the theoretical mean value and standard deviation obtained with *CSA*.

From these experiments it can be easily seen that if $\overline{p(t_i)}$ denotes the mean value of all the samples obtained with *DSA* for the computation of $p(t_i)$ then: $m_{p(t_i)} - 2\sigma_{p(t_i)} \leq \overline{p(t_i)} \leq m_{p(t_i)} + 2\sigma_{p(t_i)}$. Hence the experiments show clearly that *CSA* is a good model for *DSA* and the CESTAC method.

5. Conclusion

Starting from a minimal set of empirically known facts related to stochastic numbers, we formally deduce a number of properties and relations. We investigate the complete set of all stochastic numbers and show that this set possesses nice algebraic properties. We point out to the distinct algebraic nature of the spaces of mean-values and standard deviations. Based on the algebraic properties of the complete set of stochastic numbers we propose a natural relation for inclusion, called stochastic inclusion. A numerical example based on the Lagrange polynomial demonstrates the consistency between the CESTAC method and the presented theory of stochastic numbers.

Acknowledgments. The authors want to thank an anonymous referee for his useful comments.

References

[1] Alt, R., S. Markov, On the Algebraic Properties of Stochastic Arithmetic. Comparison to Interval Arithmetic, In: W. Kraemer and J. Wolff v. Gudenberg (Eds.), Scientific Computing, Validated Numerics, Interval Methods, Kluwer, 2001, 331–342.

[2] Alt, R., J.-L. Lamotte, S. Markov, Abstract structures in stochastic arithmetic, In: B. Bouchon-Meunier, R. R. Yager (Eds.), Proc. 11-th Conference on Information Processing and Management of Uncertainties in Knowledge-based Systems (IPMU 2006), Editions EDK, Paris, 2006, 794–801.

[3] Alt, R., J. Vignes, Validation of Results of Collocation Methods for ODEs with the CADNA Library. *Appl. Numer. Math.* 20 (1996), 1–21.

[4] Chesneaux, J. M., J. Vignes, Les fondements de l'arithmétique stochastique, *C.R. Acad. Sci.*, Paris, Sér. I, Math. 315 (1992), 1435–1440.

[5] INTLAB — INTerval LABoratory Version 5.2., www.ti3.tu-harburg.de/ rump/intlab/

[6] Markov, S., R. Alt, Stochastic arithmetic: Addition and Multiplication by Scalars, *Appl. Numer. Math.* 50 (2004), 475–488.

[7] Markov, S., R. Alt, J.-L. Lamotte, Stochastic Arithmetic: S-spaces and Some Applications, *Numer. Algorithms* 37 (1–4), 275–284, 2004.

[8] Vignes, J., R. Alt, An Efficient Stochastic Method for Round-Off Error Analysis, in: *Accurate Scientific Computations, LNCS 235*, Springer, 1985, 183–205.

[9] Vignes, J., Review on Stochastic Approach to Round-Off Error Analysis and its Applications. *Math. and Comp. in Sim.* 30, 6 (1988), 481–491.

[10] Vignes, J., A Stochastic Arithmetic for Reliable Scientific Computation, *Math. and Comp. in Sim.* 35 (1993), 233–261.

[11] Yao, Y. Y., Granular Computing: basic issues and possible solutions, In: P. P. Wang (Ed.), *Proc. of the 5-th Joint Conference on Information Sciences, Vol. I, Atlantic City, New Jersey, USA, February 27–March 3, 2000*, Association for Intelligent Machinery, 186–189.

[12] `http://www.lip6.fr/cadna`

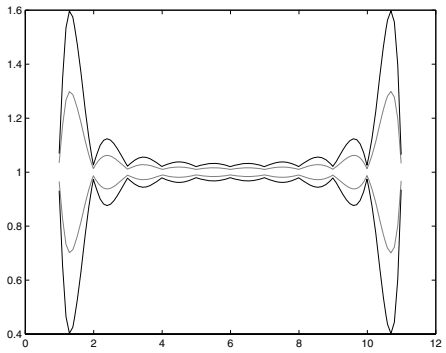


Figure 1. Lagrangian polynomial obtained with interval arithmetic (black) and CSA (gray)

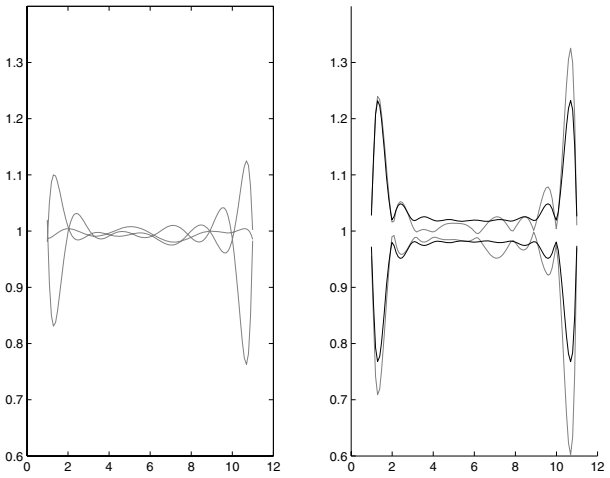


Figure 2. *DSA* **3 samples**

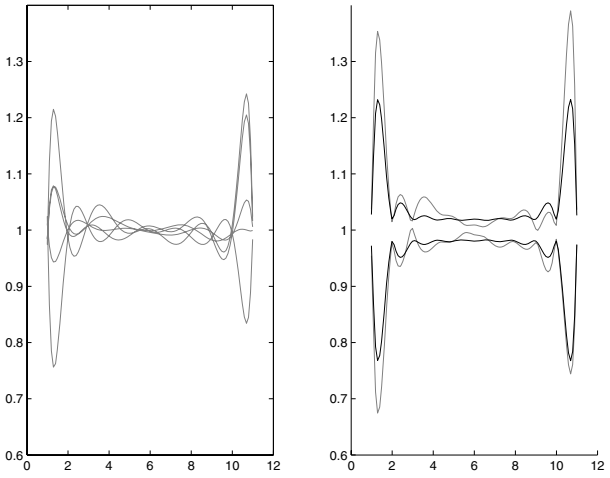


Figure 3. *DSA* **5 samples**

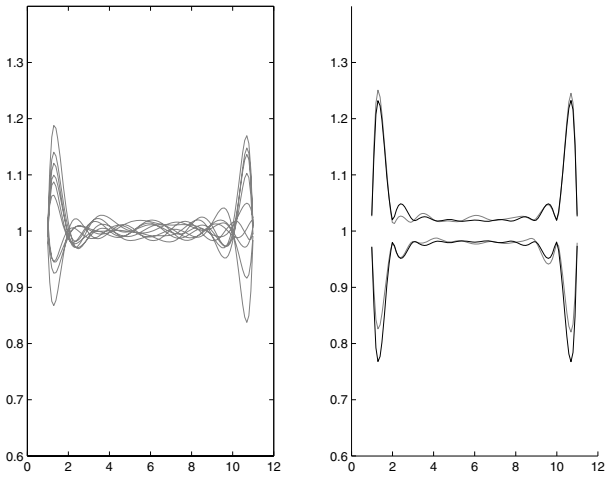


Figure 4. *DSA* **10 samples**

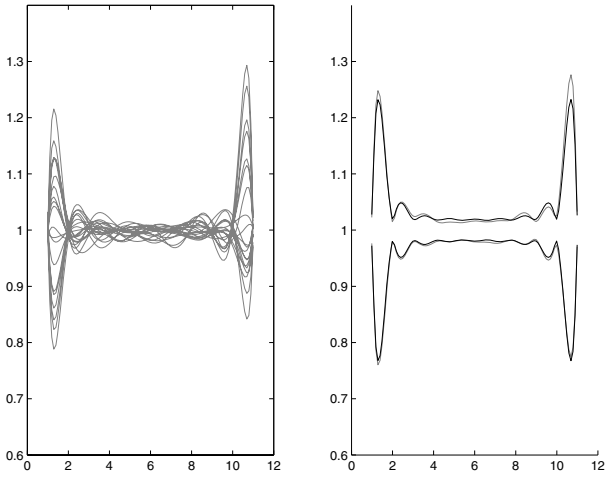


Figure 5. *DSA* **20 samples**

Toward Validating a Simplified Muscle Activation Model in SMARTMOBILE

Ekaterina Auer
Faculty of Engineering, IIIS
University of Duisburg-Essen
D-47048, Germany
auer@inf.uni-due.de

Martin Tändl, Daniel Strobach, Andres Kecskeméthy
Faculty of Engineering, IMSD
University of Duisburg-Essen
D-47048, Germany
{martin.taendl,daniel.strobach,andres.kecskemethy}
@uni-due.de

Abstract

In this paper, we apply the validated modeling and simulation environment SMARTMOBILE *along with two solvers recently added to its core to the problem of the identification of muscle activation in general motor tasks. The identification of muscle activation is one of the important and still open problems in biomechanics which aims at helping physicians assess an individual therapy for a patient. We address the discrepancy of the results supplied by the proposed simplified muscle activation model to the gait lab data and make an initial analysis of model's parameter sensitivity using validated techniques.*

1 Introduction

Modeling and simulation of kinematics and dynamics of mechanical systems is employed in many branches of modern industry and applied science. This fact contributed to the appearance of various tools for automatic generation and simulation of models of multibody systems. Possible measurement uncertainties in model parameters and errors induced by model idealization or the use of floating point numbers encourage the employment of validated arithmetics and methods in such tools.

SMARTMOBILE (Simulation and Modeling of dynAmics in MOBILE: Reliable and Template based) [1] is an environment for validated modeling and simulation of mechanical systems built on top of the non-validated MOBILE [7]. SMARTMOBILE verifies kinematics and dynamics for various classes of systems including non-autonomous and closed-loop ones. For this purpose, a basic data type (e.g. `TMoInterval` based on `INTERVAL` from PROFIL/BIAS) is combined with an appropriate validated solver (e.g. VNODE-based [13] initial value problem (IVP) solver `TMoAWAIntegrator` for dynamical tasks). Moreover, SMARTMOBILE facilitates the integration of other validated arithmetics and solvers into its core owing to its template-based, object-oriented structure. This way, a validation technique can be chosen according to the modeling task at hand, which might help to overcome overestimation or CPU time problems during simulation.

One of the possible application areas of validated methods is biomechanics. For example, the identification of muscle activation in general motor tasks can yield essential information for the physicians for therapy planning. This is a still open problem in biomechanics. Especially in the context of surgical interventions, the information about the contribution of a single muscle to joint moments during motion can enable the physician to assess a therapy before applying it to a patient. Typical solutions to this problem usually have long computation times, which make them unsuitable for on-line motion approximation required for patient-specific therapy. Recently, a fast method has been developed to roughly identify muscle activation profiles. Although suitable for first identification of activation profiles despite considerable simplifications, the approach produces results which differ from the prototype ones.

In this paper, a first study of the proposed approach from the validated point of view is made. Here, this method was simplified still further for the validation purposes which led to a relatively rough muscle activation model. An ideal goal is to detect the source of the deviation of the model results to the prototype ones. Since most of the model parameters cannot be measured exactly, and the above mentioned approach does not take the uncertainty in consideration, this might be a possible error source. On the other hand, the deviation might also result from the inadequately chosen muscle models. Considerable model simplifications performed for this first study do not allow us to make certain statements about that. Instead, we make a step in this direction by analyzing the influence of small changes in parameters on the model behavior in order to determine a critical set. This important task is especially difficult to handle with validated methods due to the chaotic character of

the considered model. For this analysis, the validated environment SMARTMOBILE and an explicit symbolic model were used.

The article is structured as follows. In Section 2, we describe the main features and recent developments of SMARTMOBILE as well as provide brief information on validated libraries it employs. Besides, a simple usage example is supplied. In Section 3, we focus on validation of the proposed simplified muscle activation model. This includes the description of the model and the comparison of the validated results for point interval parameters to the experimental data. After that, the influence of the uncertainty in several model parameters on the dynamic model behavior is studied. Finally, we recapitulate the main results from the paper and provide an outlook on future research in Section 4.

2 Main Features of SMARTMOBILE

The validated environment SMARTMOBILE is built on top of the non-validated object oriented C++ tool MOBILE for modeling and simulation of kinematics and dynamics of mechanical systems. MOBILE uses the multibody method based on the concept of a transmission element which maps motion and force between system states. A rigid link modeling rigid bodies or an elementary joint modeling revolute and prismatic joints are examples of such transmission elements. Mechanical systems are considered to be concatenations of transmission elements leading to serial chains, tree type or closed loop systems. With the help of the global kinematics, the transmission function of the complete system chain can be obtained from transmission functions of its parts. Using the inverse kinematics and the kinetostatic method [8], it is possible to build dynamic equations of motion, which are solved with common IVP solvers. MOBILE belongs to the numerical type of modeling software, that is, it does not produce a symbolic description of the resulting model. Only the values of output parameters for the user-defined values of input parameters and the source code of the program itself are available.

In SMARTMOBILE, validated arithmetics and corresponding IVP solvers are used instead of usual floating point ones. Besides, an external software for algorithmic differentiation is necessary in some cases. In this way, SMARTMOBILE can model and perform validated simulation of the behavior of mechanical systems as well as provide more realistic models by taking into account the uncertainty in parameters.

In this Section, we first list the employed validated tools and their features, and then turn to the main characteristics of SMARTMOBILE and its basic usage. At last, validated IVP solvers supplied with SMARTMOBILE are compared using a simple example. This comparison is meant to provide potential users with rudimentary criterions of choosing an appropriate solver for their problems.

2.1 Employed Validated Software

At the moment, SMARTMOBILE provides the choice between two kinds of validated arithmetics: interval and Taylor model-based. The library PROFIL/BIAS[1] is used for the former, and COSY[2] and RIOT[3] for the latter.

In the interval case, two IVP solvers are employed: VALENCIA-IVP [1] and VNODE [13]. Since both of them require derivatives of the right side of the first (the former) and higher (the latter) orders, the libraries FADBAD [2] and TADIFF [3] are used for algorithmic differentiation of the code of MOBILE, which is necessary because the symbolic model description is not available. In particular, data types `TINTERVAL` for Taylor coefficients, `FINTERVAL` for Jacobians and `TFINTERVAL` for Jacobians of Taylor coefficients are employed. In the Taylor model case, it is possible to use the C++ equivalent of COSY VI [11] by M. Kletting or its C++, object oriented version RIOT [5] by I. Eble. The algorithmic differentiation required for these solvers is handled there internally.

In VALENCIA-IVP, the interval error bounds $[R(t)]$ are sought, such that the true solution of the IVP for the given initial conditions over the given time span is contained in the enclosure $[x_{encl}(t)] = x_{app}(t) + [R(t)]$, where $x_{app}(t)$ is an arbitrary (non-validated) approximate solution. The error bounds are computed using the Picard iteration in combination with the mean-value theorem, monotonicity tests, iterative range computation, and consistency tests based on backward integration.

To use VNODE, it is necessary to discretize the time span and transform the given IVP into an autonomous one. The right side of the ODE system is assumed to be continuously differentiable up to a given order $p > 0$. First, existence and uniqueness of the solution of an IVP is proved with the help of Banach's fixed point theorem and the high order method [14] which generalizes the usual techniques based on the Picard iteration [10]. After that, a tight enclosure of the solution is computed by either the direct Taylor series algorithm, Lohner's QR-factorization algorithm, or the interval Hermite-Obreschkoff algorithm. Owing to the open structure of this object oriented solver, new algorithms can be easily added to its core.

To reduce overestimation, COSY VI expands the solution using high order Taylor series in time and initial conditions. It models the local functional behavior with the Picard iteration in combination with Schauder's fixed-point theorem. The long-term growth of integration errors is con-

[1] `www.ti3.tu-harburg.de/Software/PROFILEnglisch.html`
[2] `www.bt.pa.msu.edu/index_cosy.htm`
[3] `http://iamlasun8.mathematik.uni-karlsruhe.de/~ae08/`

trolled by the shrink wrapping method [4] and with the help of QR, blunting, and curvilinear preconditioning methods [12].

2.2 Features and Usage of SMARTMOBILE

The focus of SMARTMOBILE is to model and simulate the dynamics of mechanical systems. For this purpose, it is necessary to solve an IVP for the differential(-algebraic) equations of motion of the system model in the state space form. As already mentioned, validated IVP solvers need derivatives of the right side of these equations, which are not easy to obtain in case of MOBILE. The usual way is to make use of automatic (or algorithmic) differentiation [6], the method that is practicable but might consume a lot of CPU time in case of such a large program as MOBILE. An alternative is to obtain the derivatives from the system's mechanics. This option is not provided by MOBILE developers yet and seems to be rather difficult to algorithmize for (arbitrary) higher orders.

Algorithmic differentiation of a piece of program code can be implemented mainly through overloading or program transformation. In the first case, a new data type is developed that is capable of computing the derivative along with the function value. This new data type is used instead of the simple one in the code piece, which automatically supplies the derivative. The drawback of this method is the lack of optimization during the derivative computation. The technique of program transformation presupposes the development of a compiler that takes the original code fragment and the set of differentiation rules as its input and produces a program delivering derivatives as its output. This way might be difficult to implement for large pieces of code which are self-contained programs themselves. However, derivatives can be evaluated more efficiently with this technique.

In case of SMARTMOBILE, overloading was chosen (see Tab. 1). All relevant occurrences of `MoReal` (an alias of `double` in MOBILE) have to be replaced with an appropriate new data type. To provide interval-based validation of dynamics with the help of VNODE-based solver `TMoAWAIntegrator`, the basic data type `TMoInterval` including data types necessary for algorithmic differentiation should be used. The data type `TMoFInterval` enables the use of `TMoValenciaIntegrator`, an adjustment of the basic version of VALENCIA-IVP. The newly developed `TMoRiotIntegrator` is based on the IVP solver from the library RIOT and requires the class `TMoTaylorModel`, a SMARTMOBILE-compatible wrapper of the library's own data type `TaylorModel`. Analogously, to be able to use an adjustment of the solver COSY VI, the wrapper `RDAInterval` is necessary. Modification of the latter solver for SMARTMOBILE is currently work in progress.

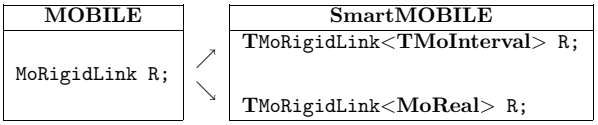


Figure 1. Template usage.

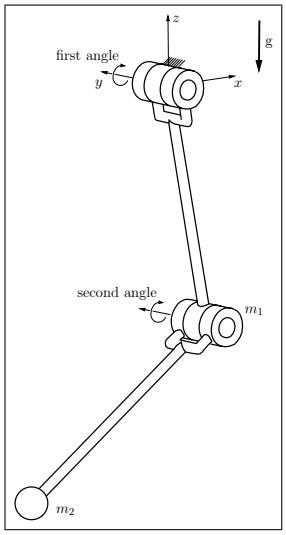


Figure 2. The double pendulum.

The availability of several basic data types in SMARTMOBILE points out its second feature: the general data type independency through its template structure. That is, `MoReal` is actually replaced not with a concrete data type, but with a placeholder. For example, the transmission element `MoRigidLink` from MOBILE is replaced with its template equivalent `TMoRigidLink`, the content of the placeholder for which (e.g. `TMoInterval` or `MoReal`, cf. Fig. 1) can be defined at the final stage of the system assembly. This allows us to use a suitable pair consisting of the data type and solver depending on the application at hand. If only a reference about the form of the solution is necessary, `MoReal` itself and a common numerical solver (e.g. Runge-Kutta's) can be used. If a relatively fast validation of dynamics without much uncertainty in parameters is of interest, `TMoInterval` and `TMoAWAIntegrator` might be the choice. For validation of highly nonlinear systems with a considerable uncertainty, the slow combination of `TMoTaylorModel` and `TMoRiOTIntegrator` can be used.

2.3 Comparison of Available Solvers

Let us consider the example of the double pendulum with an uncertain initial angle of the first joint from [1], shown in Fig. 2. We study the dynamics of the double

Table 1. Basic validated data types and the corresponding solvers in SMARTMOBILE.

Data type	Solver	Required tools
`class TMoInterval{ INTERVAL Enclosure; TINTERVAL TEnclosure; TFINTERVAL TFEnclosure;}`	`TMoAWAIntegrator`	PROFIL/BIAS, FADBAD, TADIFF
`TMoFInterval= {double,INTERVAL,FINTERVAL}`	`TMoValenciaIntegrator`	PROFIL/BIAS, FADBAD, VALENCIA-IVP
`TMoTaylorModel={TaylorModel}`	`TMoRiOTIntegrator`	RIOT
`RDAInterval={Cosy}`	COSY VI–based solver	COSY, COSY VI

```
MoFrame K0, K1, K2, K3, K4;
MoAngularVariable psi1, psi2;
// transmission elements
MoVector l1(0,0,-1), l2(0,0,-1) ;
MoElementaryJoint R1(K0,K1,psi1,xAxis) ;
MoElementaryJoint R2(K2,K3,psi2,xAxis) ;
MoRigidLink rod1(K1,K2,l1),rod2(K3,K4,l2) ;
MoReal m1(1),m2(1) ;
MoMassElement Tip1(K2,m1),Tip2(K4,m2) ;
// the complete system
MoMapChain Pend;
Pend << R1<<rod1<<Tip1<<R2<<rod2<<Tip2 ;
// dynamics
MoVariableList q; q << psi1<<psi2 ;
MoMechanicalSystem S(q,Pend,K0,zAxis) ;
MoAdamsIntegrator I(S) ;
for(int i=0;i<100;i++) I.doMotion();
```

```
#define TMoInterval t;
TMoFrame<t> K0, K1, K2, K3, K4;
TMoAngularVariable<t> psi1, psi2;
// transmission elements
TMoVector<t> l1(0,0,-1), l2(0,0,-1) ;
TMoElementaryJoint<t> R1(K0,K1,psi1,xAxis) ;
TMoElementaryJoint<t> R2(K2,K3,psi2,xAxis) ;
TMoRigidLink<t> rod1(K1,K2,l1),rod2(K3,K4,l2) ;
t m1(1),m2(1) ;
TMoMassElement<t> Tip1(K2,m1),Tip2(K4,m2) ;
// the complete system
TMoMapChain<t> Pend;
Pend << R1<<rod1<<Tip1<<R2<<rod2<<Tip2 ;
// dynamics
TMoVariableList<t> q; q << psi1<<psi2 ;
TMoMechanicalSystem<t> S(q,Pend,K0,zAxis) ;
TMoAWAIntegrator I(S,0.0001,ITS_QR,15) ;
I.doMotion();
```

Figure 3. The double pendulum in MOBILE (left) and SMARTMOBILE (right).

pendulum using a SMARTMOBILE model from Fig. 3, where it is shown in comparison to the corresponding model from MOBILE. Note that MOBILE users can switch to SMARTMOBILE easily since the difference between the models is minimal. The lengths of both massless arms of the pendulum are equal to $1\ \mathrm{m}$ and the two point masses amount to $1\ \mathrm{kg}$ each with the gravitational constant $g = 9.81\frac{\mathrm{m}}{\mathrm{s}^2}$. The initial values for angles (specified in rad) and angular velocities (in $\frac{\mathrm{rad}}{\mathrm{s}}$) are given as

$$\underline{\beta}^0 = \left[0.99\frac{3\pi}{4} \quad -\frac{11\pi}{20} \quad 0.43 \quad 0.67\right]^T,$$

$$\overline{\beta}^0 = \left[1.01\frac{3\pi}{4} \quad -\frac{11\pi}{20} \quad 0.43 \quad 0.67\right]^T,$$

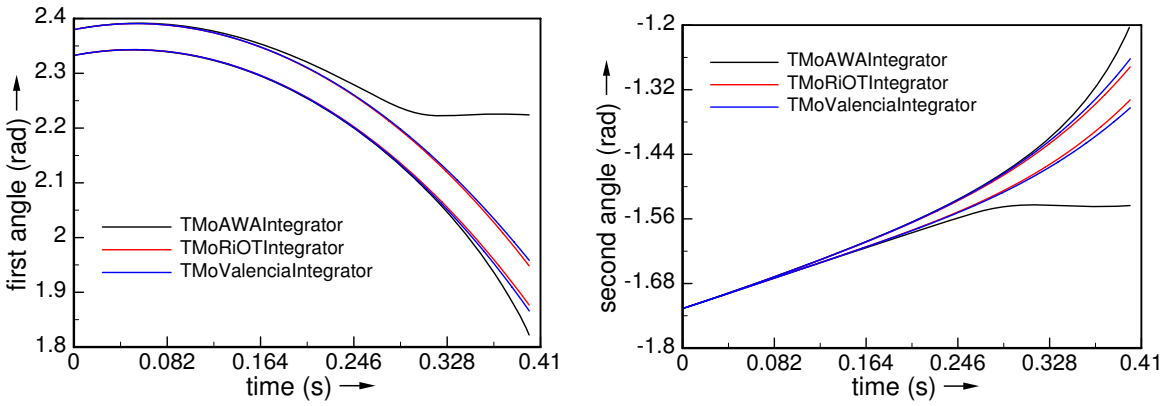
where the initial angle of the first joint has an uncertainty of ± 1 percent of its nominal value.

The results are summarized in Tab. 2. In case of `TMoAWAIntegrator`, the QR-factorization algorithm with Taylor series of order 15 and step size $h = 0.0001$ is chosen. `TMoValenciaIntegrator` has a step size $h = 0.0001$, too. `TMoRiotIntegrator` is used with Taylor models of order 5, without the shrink wrapping, and with a variable step size $0.0002 \leq h \leq 0.2$. The line "Break-down" of Tab. 2 contains the time in seconds after which the corresponding method no longer works. The last line indicates the CPU time (in seconds) which the solvers take to obtain the solution over the integration interval $[0; 0.4]$ on a Pentium 4, 3.0 GHz PC using CYGWIN. Note that the CPU times are provided only as a rough reference since the solvers can be further optimized in this respect. Additionally, the interval enclosures of the two angles β_1 and β_2 of the double pendulum are shown for identical time intervals in Fig. 4.

The use of `TMoValenciaIntegrator` improves both the tightness of the resulting enclosures and the CPU time in comparison to `TMoAWAIntegrator` for this ex-

Table 2. Performance of `TMoAWAIntegrator`, `TMoRiOTIntegrator`, and `TMoValenciaIntegrator` for the double pendulum over the time interval $[0; 0.4]$.

Strategy	`TMoAWAIntegrator` ($h = 0.0001$)	`TMoRiOTIntegrator` ($0.0002 \leq h \leq 0.2$)	`TMoValenciaIntegrator` ($h = 0.0001$)
Break–down	0.424	0.820	0.504
CPU Time	1248	9312	294



(a) Enclosure of the first joint angle.

(b) Enclosure of the second joint angle.

Figure 4. Interval enclosures for the first and second state variable of the double pendulum.

ample. Although `TMoRiOTIntegrator` breaks down much later than the both former solvers, it needs a lot of CPU time (cf. Tab. 2).

3 A Simplified Muscle Activation Model

In this Section, we first describe the simplified muscle activation model under consideration. Then we compare the validated results for the model without uncertainty to the results from the gait lab. After that, SMARTMOBILE and a corresponding symbolic model in combination with a validated IVP solver are used to investigate the influence of small changes in the chosen parameters on the simulation.

3.1 Description

The model under investigation represents a simplified subsystem of the human leg (cf. Fig. 5). It consists of pelvis, thigh and shank, represented by the corresponding bones *hipbone*, *femur* and *tibia*. The two joints at hip and knee limit the possible range of motion to the sagittal plane. To drive the model in forward dynamics simulations, the muscle *biceps femoris short head* is included, which is responsible for knee flexion. Segment dimensions, mass properties and characteristic muscle parameters (origin and

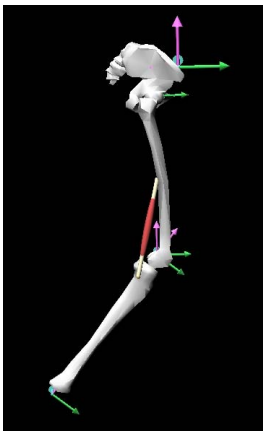


Figure 5. A simplified leg model in MOBILE.

insertion points, actuator reference length, ratio of tendon length and muscle tissue length in the actuator) stem from a male adult of 1,78 m body height and a weight of 77,9 kg. All used data is taken from [16]. To enable reasonable computation times in forward dynamics simulations, the force law of the involved muscle model is roughly simplified with respect to the HILL-type muscle model described in [17].

The activation of the muscle is modeled using linear

combinations of exponential functions to reduce the number of parameters for each muscle. The method is described in detail in [15]. For the purposes of this first validated study, the overall function was simplified so that it is everywhere continuously differentiable. We chose a muscle activation such that the motion (hip and knee angle) produced by the simulation approximates measurements performed in a gait lab. In this setting, the target motion represents a flexion-extension movement of the knee, starting and ending in a stretched leg position (hip and knee flexion angle $\approx 0^\circ$).

The full MOBILE code of the studied model can be made available by the authors on demand. The validated model from SMARTMOBILE is similar to that of MOBILE. The model behavior is studied using the solvers `TMoAWAIntegrator` (Lohner's QR-factorization algorithm with order 15 and a variable step size), `TMoValenciaIntegrator` (with the step size 0.0002) and `TMoRiOTIntegrator` (with order 5, a variable step size and without shrink wrapping) described in Section 2.3.

On the other hand, a corresponding *symbolical* model can be derived using MOMAPLE[4], a rudimentary plug-in for MOBILE capable of modeling only tree-type mechanisms. Through a program very similar to that of MOBILE, a MAPLE worksheet producing the symbolic form of equations of motion is automatically generated. Here, the equations can be symbolically simplified if all the parameters are assigned their values *before* the function for the right side of the system is evaluated. However, the explicit dependence on parameters is then lost. We made a compromise and let the symbolical model explicitly depend only on the thigh length and the coordinates of the proximal and distal insertion points for the muscle.[5] The equations can be solved by a validated IVP solver. We chose VNODE-2.0 (Lohner's QR-factorization algorithm with order 15 and a variable step size) and COSY VI version of November 10, 2006 (with (weighted) order of 12, a variable step size and without preconditioning and shrink wrapping).

3.2 Model Analysis

In the above model, there exist many parameters that cannot be measured exactly. They are, for example, the lengths of the thigh and shank as well as coordinates of muscle insertion points. If the model is simulated in SMARTMOBILE with point interval parameters, a considerable discrepancy between the gait lab and model data can be observed. In Fig. 6, the dependence of `TMoAWAIntegrator` enclosures of hip and knee angles on time is shown in comparison to the gait lab data for point interval parameters. The source of the discrepancy might be inexact measurements which influence both the gait lab data and the values of parameters from literature used in the muscle activation model. On the other hand, the idealization and simplification of the model itself contribute to that.

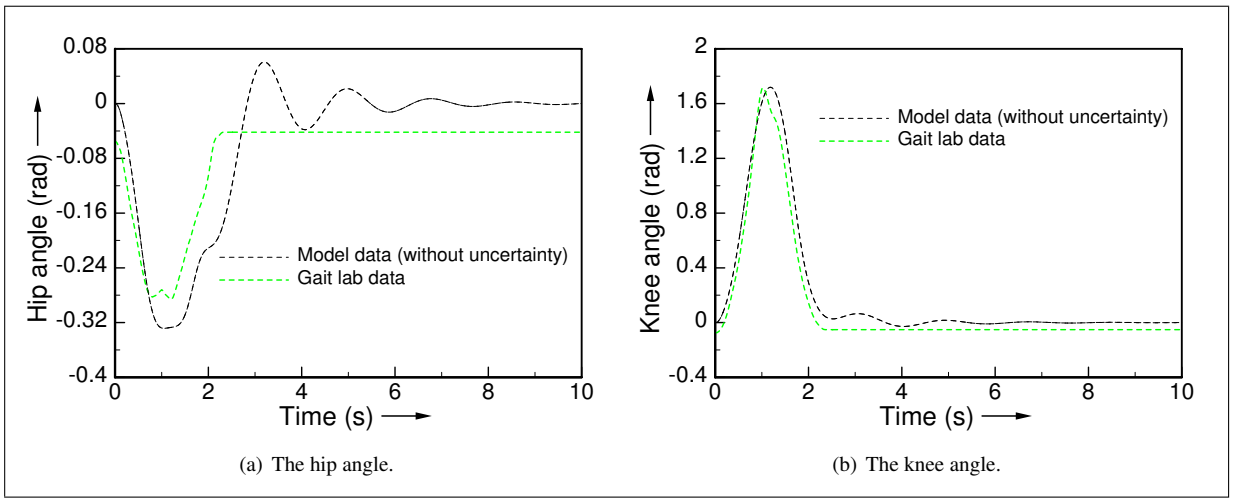
The simulation results for the example from Section 2.3 imply that validated methods are able to produce meaningful results only over short integration periods if the parameters are chosen to be intervals. This also concerns the simple biomechanical model used here because it has a double pendulum as its basis (cf. Figures 2 and 5). In Fig. 7, this suspicion is confirmed. The Figure shows the influence of ± 0.1 percent uncertainty in the thigh length on the behavior of the numerical (left) and symbolic (right) model. The integration process can reach at most $t = 0.54337$ seconds (for the more simple symbolic model with the help of COSY VI). Moreover, it can be seen that this relatively small uncertainty (from the mechanical point of view) cannot account for the discrepancy between the model and lab data. Although for several obtained enclosures the empirical results seem to lie within validated bounds, the comparison with better bounds show that this is rather the result of overestimation of the corresponding method and chaotic character of the model.

From the mechanical point of view, the coordinates of insertion points of the muscles seem to be especially difficult to measure exactly. Besides, most of the literature data is provided for a human of a certain height and weight (cf. Section 3.1), which brings still more uncertainty in to the model for a concrete patient. Because of that, it is especially important to find the set of sensitive parameters. In Tab. 3, our first validated parameter analysis is summarized. We restricted the set of parameters to the thigh and shank lengths as well as the coordinates of the proximal insertion point for the *biceps femoris short head*. We introduced one percent and one tenth of percent uncertainty in each parameter. The goal was to compare the break down times for these uncertainties in the numerical and more simple symbolic case (with the help of `TMoAWAIntegrator` and VNODE, respectively).

The analysis of both models shows that the most sensitive parameters are the thigh length, the shank length and the z coordinate of the proximal muscle insertion point. (The shank length was not studied in the symbolic case owing to conventions mentioned at the end of the Section 3.1.) Besides, the x coordinate has a remarkable influence despite its relatively small value. In Tab. 4, we study the the thigh length in more detail with the help of the symbolic model and VNODE. (Reference break down time means that only a reference is available here for this characteristic since the simulation was manually aborted as soon as the width of the enclosures was larger than of order 10.) It can be observed

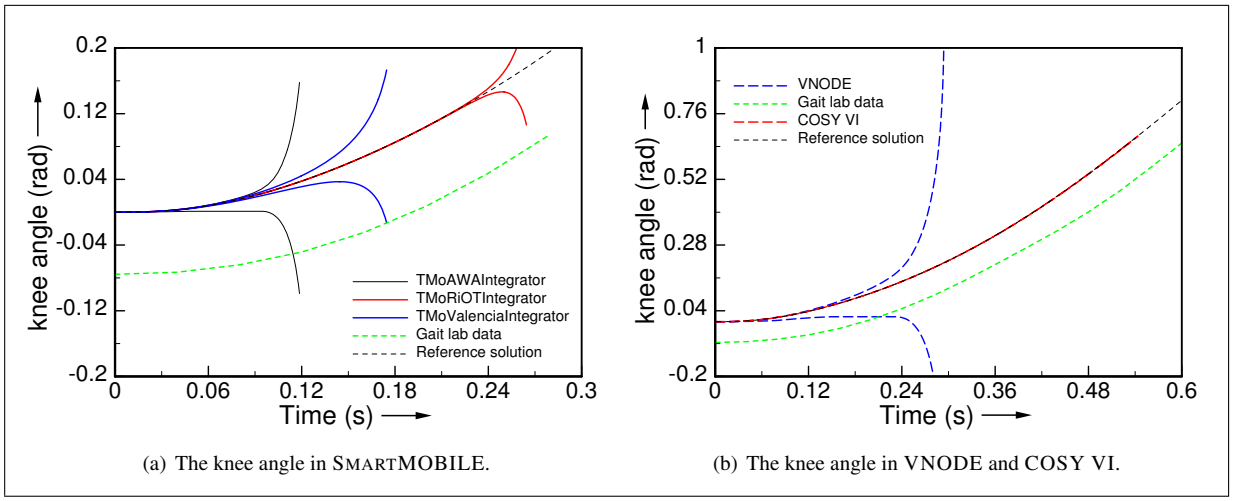
[4] Release of September 2006 by M.Tändl, e-mail `martin.taendl@uni-due.de`

[5] Note that the numerical model from SMARTMOBILE cannot be simplified in this way and is therefore more complex.



(a) The hip angle. (b) The knee angle.

Figure 6. Interval enclosures and and gait lab data for the hip and knee angles (no uncertainty).



(a) The knee angle in SMARTMOBILE. (b) The knee angle in VNODE and COSY VI.

Figure 7. Interval enclosures of the knee angle under ± 0.1 percent uncertainty in the thigh length.

that this parameter should be measured exactly up to the order of a micrometer to have little influence on the system in the framework of VNODE.

Table 4. Influence of the uncertainty in the thigh length.

Uncertainty	Reference break down time (s)
$\pm 0.001\%$	$4.2e-01$
$\pm 0.0001\%$	$7.0e-01$
$\pm 0.00001\%$	$> 1.0e+00$

4 Conclusions and Outlook on Future Research

In this paper, we presented two recently developed solvers for the validated modeling and simulation environment SMARTMOBILE: `TMoValenciaIntegrator` based on VALENCIA-IVP and `TMoRiOTIntegrator` based on `RiOT`. The full range of SMARTMOBILE solvers was applied to analyze the simplified muscle activation model which aimed at helping physicians assess an individual patient therapy. Due to the further model simplification for validation purposes, the question about the source of the discrepancy of the model results and the gait lab data could not be answered unambiguously. However, first steps in this direction were made and the set of especially sensitive parameters of the model identified. Generally, the task

Table 3. Influence of the parameters on the simplified muscle activation model

Parameter	Break down (`TMoAWAIntegrator`)		Break down (VNODE)	
	$\pm 1\%$	$\pm 0.1\%$	$\pm 1\%$	$\pm 0.1\%$
Thigh length $= 0.45$	$8.4068249e-02$	$1.1859999e-01$	$1.5557e-01$	$2.6935e-01$
Shank length $= 0.49$	$8.1839977e-02$	$1.2529627e-01$	—	—
IP, coordinate $z = -0.2281$	$9.4212640e-02$	$1.3543401e-01$	$2.0333e-01$	$3.7424e-01$
IP, coordinate $y = -0.0253$	$2.5418879e-01$	$3.8887304e-01$	$6.5233e-01$	$1.6692e+00$
IP, coordinate $x = 0.0054$	$1.1982085e-01$	$1.8389055e-01$	$4.2507e-01$	$8.0971e-01$

of taking into consideration all uncertainties that can appear in this model in a validated way seems to be a difficult one due to the chaotic nature of the underlying system of equations.

Still, further steps toward this goal are planned. First, we will consider the model without simplifications which were made for validation. This will help better approximate the gait lab data and so allow us to choose smaller uncertainties. This might involve the necessity to differentiate piecewise continuously differentiable functions automatically in SMARTMOBILE. Alternatively, such functions can be artificially made differentiable. Both of these approaches involve considerable effort on the side of MOBILE and SMARTMOBILE developers.

A second direction of our future work will be to further reduce the overestimation in SMARTMOBILE. On the one hand, this will include the adjustment of such solvers as VSPODE [9] by Stadtherr and Lin to SMARTMOBILE. On the other hand, the consideration of splitting and merging techniques proposed by the group around E.P. Hofer in combination with backward integration and parallelization seems promising.

References

[1] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SMARTMOBILE: Improvement of Performance by VALENCIA-IVP. In *Proc. of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice*, Lecture Notes in Computer Science, 2006. To appear.

[2] C. Bendsten and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods. Technical Report 1996-x5-94, Technical University of Denmark, Lyngby, 1996.

[3] C. Bendsten and O. Stauning. TADIFF, a flexible C++ package for automatic differentiation using Taylor series. Technical Report 1997-x5-94, Technical University of Denmark, Lyngby, 1997.

[4] M. Berz and K. Makino. Suppression of the Wrapping Effect by Taylor Model-Based Verified Integrators: Long-term Stabilization by Shrink Wrapping. *International Journal of Differential Equations and Applications*, 2006. In print. Online.

[5] I. Eble. RiOT. Available through the author: `http://iamlasun8.mathematik.uni-karlsruhe.de/~ae08/`.

[6] A. Griewank. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2000.

[7] A. Kecskeméthy. *Objektorientierte Modellierung der Dynamik von Mehrkörpersystemen mit Hilfe von Übertragungselementen*. PhD thesis, Gerhard Mercator Universität Duisburg, 1993.

[8] A. Kecskeméthy and M. Hiller. An object-oriented approach for an effective formulation of multibody dynamics. *Computer Methods in Applied Mechanics and Engineering*, 115:287–314, 1994.

[9] Y. Lin and M. A. Stadtherr. Validated solution of initial value problems for ODEs with interval parameters. In *NSF Workshop Proceeding on Reliable Engineering Computing*, Savannah GA, February 22-24, 2006.

[10] R. Lohner. *Einschließung der Lösung gewönlicher Anfangs- und Randwertaufgaben und Anwendungen*. PhD thesis, Universität Karlsruhe, 1988.

[11] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, 1998.

[12] K. Makino and M. Berz. Suppression of the Wrapping Effect by Taylor Model-Based Verified Integrators: Long-term Stabilization by Preconditioning. *International Journal of Differential Equations and Applications*, 2006. In print. Online.

[13] N. S. Nedialkov. *The design and implementation of an object-oriented validated ODE solver*. Kluwer Academic Publishers, 2002.

[14] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001.

[15] D. Strobach, A. Kecskeméthy, G. Steinwender, and B. Zwick. A simplified approach for rough identification of muscle activation profiles via optimization and smooth profile patches. In *CD Proceedings of the International ECCOMAS Thematic Conference on Advances in Computational Multibody Dynamics*, Madrid, Spain, June 21 – 24 2005. ECCOMAS.

[16] G. T. Yamaguchi. *Dynamic modeling of musculoskeletal motion*. Kluwer Academic Publishers, 2001.

[17] F. E. Zajac. *Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control*. 1989.

GRKLib: a Guaranteed Runge Kutta Library

Olivier Bouissou
CEA LIST
olivier.bouissou@cea.fr

Matthieu Martel
CEA LIST
matthieu.martel@cea.fr

Abstract

In this article, we describe a new library for computing guaranteed bounds of the solutions of Initial Value Problems (IVP). Given an initial value problem and an end point, our library computes a sequence of approximation points together with a sequence of approximation errors such that the distance to the true solution of the IVP is below these error terms at each approximation point. These sequences are computed using a classical Runge-Kutta method for which truncation and roundoff errors may be over-approximated. We also compute the propagation of local errors to obtain an enclosure of the global error at each computation step. These techniques are implemented in a C++ library which provides an easy-to-use framework for the rigorous approximation of IVP. This library implements an error control technique based on step size reduction in order to reach a certain tolerance on local errors.

1. Introduction

Users of numerical solvers for Ordinary Differential Equations (ODEs) are generally interested in computing approximation points with an *estimate* of the error at each point. Many scientists and engineers are more interested in the efficiency of the method than on the quality of the error estimation. However, for many safety critical applications, this estimation is not enough, and safe bounds on the error are required. These applications that need *guaranteed* approximations include state estimation [8], hybrid systems analysis [6] or industrial systems where critical damage may occur. For such systems, the required feature of the solver is safety rather than efficiency, i.e. the numerical solver should not only give an approximation of the solution but it should also prove that the true solution lies between computed bounds. This problem has been studied over the past 40 years, i.e. almost since interval arithmetic was invented. However, using interval versions of classical algorithm gives validated solvers which usually suffer from a bad long term stability. The most successful methods are based on a Taylor Series expansion of the solution with respect to time and a fine algorithm for computing the remainder terms.

In the rest of this introduction, we briefly recall what an Initial Value Problem is and we give the foundations of Taylor Series based methods for computing rigorous bounds of IVPs. For a more complete description of these methods and tools implementing them, we invite the reader to refer to [13].

1.1. Initial Value Problems

An IVP consists of a system of ODEs together with an initial condition:

$$\dot{y} = f(y) \qquad y(x_0) = y_0. \tag{1.1}$$

Here, y is a function from $\mathbb{R}$ to $\mathbb{R}^n$ and f is a continuous function from $\mathbb{R}^n$ to $\mathbb{R}^n$. $\dot{y}$ denotes the derivative of y with respect to time x.

Solving the IVP means finding a (possibly unique) function $y(x; x_0, y_0)$ which satisfies Equation (1.1). Numerical solvers usually compute a sequence of approximation points $\{y_0, y_1, \ldots, y_M\}$ such that y_i is an approximation of the value $y(x_i; x_0, y_0)$, for some x_i. Let $y_i^v = y(x_i; x_0, y_0)$ be the real value of the solution of (1.1) at time x_i. The sequence $\big(x_n\big)$ is the sequence of *steps*, and we let $h_i = x_{i+1} - x_i$ denote the step sizes. In sections 2 and 3, we assume a fixed step size. The step size control mechanism and its influence on the algorithm are detailed in Section 4.

If we consider systems with uncertainties, the initial conditions are not always exactly known, or we may only have approximate values for the parameters of the equations. Therefore, we will focus on a more general IVP, where the initial conditions are given as follows:

$$y(x_0) \in [y_0], \;\; [y_0] \subseteq \mathbb{R}^n \tag{1.2}$$

Solving this interval IVP means finding the set of functions $y(x; x_0, [y_0]) = \{y(x; x_0, y_0) \mid y_0 \in [y_0]\}$. Again, this problem is difficult. All we can do is to compute a sequence of boxes $[y_n]$ such that $\forall n,\; y(x_n; x_0, [y_0]) \subseteq [y_n]$.

1.2. Taylor Series Methods in a Nutshell

The method studied and used most often to achieve guaranteed bounds on the solutions of IVPs is based on an interval version of classical Taylor Series algorithm [4, 10, 12, 17]. This method performs a Taylor decomposition of the solution of (1.1) with respect to time, in such a way that:

$$y_{j+1} = y_j + \sum_{k=1}^{N-1} f^{[k]}(y_j)h_j^k + h_j^N \cdot f^{[N]}\big(y(x_s)\big), \qquad (1.3)$$

where $x_s \in [x_j, x_{j+1}]$ and $f^{[k]}(y) = \frac{1}{k}\left(\frac{\partial f^{[k-1]}}{\partial y} f\right)(y)$, $f^{[0]}(y) = y$. A direct translation of (1.3) into interval arithmetic gives Equation (1.4), where $[\tilde{y_j}]$ is an a priori enclosure of $y(x_s)$:

$$[y_{j+1}] = [y_j] + \sum_{k=1}^{N-1} f^{[k]}([y_j])h_j^k + f^{[N]}\big([\tilde{y_j}]\big). \qquad (1.4)$$

In a direct evaluation of (1.4), the width of $[y_j]$ grows, even if the system contracts. Thus, it is important to compute $[y_{j+1}]$ in a way which limits the overestimation inherent in interval arithmetic. This is achieved by expressing the interval valued evaluations by their mean value form:

$$f(\{a \in [\underline{a}, \overline{a}]\}) \subseteq \hat{a} + J(f, [\underline{a}, \overline{a}]) \cdot ([\underline{a}, \overline{a}] - \hat{a}) \subseteq f([\underline{a}, \overline{a}]),$$

where $J(f, [\underline{a}, \overline{a}])$ is the Jacobian of the function f evaluated on the whole interval $[\underline{a}, \overline{a}]$, and $\hat{a} \in [\underline{a}, \overline{a}]$. If we apply this formula to (1.4), we obtain:

$$\begin{aligned}[y_{j+1}] \quad = \quad & \hat{y_j} + \sum_{k=1}^{N-1} f^{[k]}(\hat{y_j})h_j^k + f^{[N]}\big([\tilde{y_j}]\big) + \\ & \big(I + \sum_{k=1}^{N-1} J(f^{[k]}, [y_j])h_j^k\big)([y_j] - \hat{y_j}).\end{aligned}$$

There are still two problems to solve to use this formula:

- finding the a priori enclosure $[\tilde{y_j}]$, i.e. a box such that $\forall x \in [x_j, x_{j+1}], y(x; x_j, [y_j]) \in [\tilde{y_j}]$.
- reducing the wrapping effect when computing $\big(I + \sum_{k=1}^{N-1} J(f, [y_j])h_j^k\big)([y_j] - \hat{y_j})$.

Thus, Taylor Series methods are generally two-step methods: they first compute an a priori enclosure of the solution on one integration step, then they reduce this enclosure to get $[y_{j+1}]$ as tight as possible. We will face these two problems in our method, as developed in sections 3.1 and 3.2.

1.3. Description of our Method

The main contribution of this article is to show the feasibility of another way for computing rigorous bounds on the solution of an IVP. Our approach is comparable to the one taken by ValEncIA-IVP [15]; we compute a sequence of non-validated approximation points together with a sequence of guaranteed bounds on the distance between these points and the exact solution. Therefore, this method may be seen as a predictor-corrector algorithm: we predict the value of the approximation points, and we correct them by computing an over-approximation of the global error. The interest of this approach is that the use of interval arithmetic is limited to verification tasks (computation of the errors), thus limiting the size of the intervals to grow too much.

We chose to base our method on a classical Runge-Kutta algorithm for the computation of the approximation points. The error is then estimated using the higher order derivatives of the function y we want to approximate. It is computed as the sum of three terms: the error due to the limited order of the Runge-Kutta method, the error propagated by the dynamical system, and finally the error due to the implementation of the algorithm on a finite precision machine. We start with a brief review of the RK4 algorithm we use (Section 2). Then we show how we compute the over-approximation of the global error (Section 3). Finally, we see how the step size may be controlled to achieve a required error bound in Section 4 and we give some numerical results and benchmarks in Section 5.

2. The RK4 Algorithm

The RK4 algorithm is a Runge-Kutta algorithm of order 4. Runge-Kutta algorithms are implicit schemes which compute the sequence of approximation points $\big(y_n\big)$ using only y_n and some intermediary points for the computation of y_{n+1}. Details on these methods may be found in many numerical analysis books, for example [3, 18]. The RK4 method can be seen as an extension of Euler's ($y_{n+1} = y_n + h \cdot f(y_n)$) and Midpoint's method ($y_{n+1} = y_n + h \cdot f(y_n + h/2 \cdot f(y_n))$). It uses four evaluations of f to compute y_{n+1}: one at the beginning of the interval, two at the middle and on at the end:

$$\begin{aligned} k_1 &= f(y_n) \\ k_2 &= f(y_n + h/2 \cdot k_1) \\ k_3 &= f(y_n + h/2 \cdot k_2) \\ k_4 &= f(y_n + h \cdot k_3) \\ y_{n+1} &= y_n + \tfrac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right). \end{aligned} \qquad (2.1)$$

It is commonly known, as we will see it in the next section, that using these formulas gives an approximation of order 4, i.e. the difference between the exact value $y(x_n + h)$ and y_{n+1} is of the same magnitude as h^5. If we apply successively these formulas, we obtain a set of approximation points y_j for every $x_j = x_0 + j * h$. These points are often believed to be a good approximation of the real solution. We show in the next section how one can compute guaranteed bounds on the error $y(x_j) - y_j$ for every j. Let us define some functions which will help to express the error bounds. We make the k_i coefficients depend on y and h, and define

the iteration function Φ:

$$\begin{array}{rcl} k_1(y,h) & = & f(y) \\ k_2(y,h) & = & f(y+h/2\cdot k_1(y,h)) \\ k_3(y,h) & = & f(y+h/2\cdot k_2(y,h)) \\ k_4(y,h) & = & f(y+h\cdot k_3(y,h)) \\ \Phi(y,h) & = & y+\frac{h}{6}\big(k_1+2k_2+2k_3+k_4\big)(y,h) \end{array} \quad (2.2)$$

such that we have $y_j = \Phi^j\big(y_0,h\big)$, where Φ^j is the jth iterate of the function Φ. We also define two partial functions ψ_j and ϕ_j at every step:

$$\psi_j : y \mapsto \Phi(y,h_j) \qquad \phi_j : x \mapsto \Phi\big(y_j^v, x-x_j\big). \quad (2.3)$$

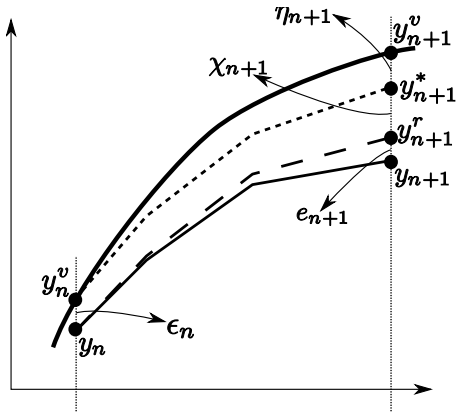


Figure 1. Three kinds of errors.

3. Computing the Error

Let us now focus on the central part of our algorithm, i.e. the computation of guaranteed bounds on the global error. Our goal is to compute upper bounds of $\big(y(x_j;x_0,y_0)-y_j\big)$ for all the approximation points. Hence, we need to address the following questions: what is the error introduced by the method and how is it propagated by the dynamical system from one step into another? Furthermore, we need to be very careful on the implementation of the method, as any roundoff error must be taken into consideration.

Let us consider Step $n+1$: we already computed y_n and $[\epsilon_n]$ at x_n such that $y_n^v \subseteq y_n + [\epsilon_n]$. We recall that $y_n^v = y(x_n;x_0,y_0)$ is the value of the solution at time x_n. Let y^r_{n+1} be the real valued point given by the Runge-Kutta fomulas, which approximates y^v_{n+1}, and let y_{n+1} be the corresponding floating point given by the implementation. This is the point we will actually use. A first source of error comes from the difference between y_{n+1} and y^r_{n+1}. In addition, there is the error propagated by the dynamical system itself. Let y^*_{n+1} be the real valued point that we would have computed if we had applied the RK4 formulas starting from y^v_n. The distance between y^r_{n+1} and y^*_{n+1} represents how $[\epsilon_n]$ has been propagated into $[\epsilon_{n+1}]$. Finally, there is the error introduced by the method itself, which is the distance between y^*_{n+1} and y^v_{n+1}. So, the global error at x_{n+1} may be decomposed into three kinds of errors (see Figure 1):

- truncation errors due to the method: $\eta_{n+1} = y^v_{n+1} - y^*_{n+1}$,
- propagation of the previous error due to the dynamical system: $\chi_{n+1} = y^*_{n+1} - y^r_{n+1}$,
- roundoff errors due to finite precision computations: $\epsilon_{n+1} = y^r_{n+1} - y_{n+1}$.

In the following, we explain how these terms are computed.

3.1. Truncation Errors

Truncation errors arise because the trajectory of the true solution to Equation (1.1) (curved line on Figure 1) and the trajectory given by the RK4 formulas (dotted line) differ. We suppose that they have a common starting point y^v_n at x_n, and we try to over-approximate their difference $y(x_{n+1};x_n,y^v_n) - y^*_{n+1}$ at $x_{n+1} = x_n + h_n$. The following proposition then holds:

Proposition 3.1 *$y^*_{n+1} = \phi_n\big(x_{n+1}\big)$, and the four first derivatives of y and ϕ_n are equal at $x = x_n$:*

$$\forall i \in [0,4],\ \frac{d^i y}{dx^i}(x_n) = \frac{d^i \phi_n}{dx^i}(x_n).$$

The proof of this proposition is straightforward and may be read in [2]. Proposition 3.2 follows immediately:

Proposition 3.2 *There exists some $\xi \in [x_n, x_{n+1}]$ such that*

$$\eta_{n+1} = \frac{h^5}{120}\frac{d^5(y-\phi_n)}{dx^5}(\xi).$$

The proof is once again a straightforward application of the Taylor Series theorem with Lagrange remainder. Now, let us compute this fifth derivative. We have:

$$\begin{array}{rcl} \dfrac{d^5(y-\phi_n)}{dx^5} & = & \dfrac{d^5(y)}{dx^5} - \dfrac{d^5(\phi_n)}{dx^5} = \dfrac{d^4(f)}{dx^4} - \dfrac{d^5(\phi_n)}{dx^5} \\ \eta_{n+1} & = & \dfrac{d^4(f)}{dx^4}\big(y(\xi;x_n,y^v_n)\big) - \dfrac{d^5(\phi_n)}{dx^5}(\xi). \end{array}$$

The function ϕ_n only depends on x; so does $\frac{d^5(\phi_n)}{dx^5}$. Therefore, we have:

$$\frac{d^5(\phi_n)}{dx^5}(\xi) \in \frac{d^5(\phi_n)}{dx^5}([x_n, x_{n+1}]). \quad (3.1)$$

For the term $\frac{d^4(f)}{dx^4}\big(y(\xi;x_n,y^v_n)\big)$, the situation is a bit more complicated, as we cannot easily enclose the value

$y(\xi; x_n, y_n^v)$. We thus need an a priori enclosure of the function y on the interval $[x_n, x_{n+1}]$, i.e. a box $[\tilde{y_n}]$ such that $\forall x \in [x_n, x_{n+1}], y(x; x_n, y_n^v) \in [\tilde{y_n}]$. This will be done using Picard operator and Banach fixed point theorem, as in [10, 12]. Let us recall the definition of the operator and the main result that we will use (Proposition 3.3).

Defintion 3.1 *Given an ODE $\dot{y} = f(y)$, $y(x_n) = y_n^v$, the associated Picard-Lindelöf operator is defined by:*

$$T(y)(t) = y_n^v + \int_{x_n}^{t} f(y(s))\, ds$$

Furthermore, let S be a closed subset of $C^0\big([x_n, x_{n+1}], \mathbb{R}^d\big)$, the set of continuous functions from $[x_n, x_{n+1}]$ into $\mathbb{R}^d$.

Proposition 3.3 *If f satisfies a Lipschitz condition and if S is mapped into itself by T, then there exists a unique solution to the ODE on $[x_n, x_{n+1}]$.*

This proposition is used to find an a priori enclosure of the solution of the ODE over the step $[x_n, x_{n+1}]$ [17]. Let $S' = \big\{y | y \in C^0\big([x_n, x_{n+1}], B\big)\big\}$ be the set of continuous functions over $[x_n, x_{n+1}]$ with value in the box B. For any $y \in S'$, we have:

$$\begin{aligned}(Ty)(t) &= y_n^v + \int_{x_n}^{t} f(y(s))ds \\ &\subseteq y_n^v + [0, h_n] \cdot f(B)\end{aligned}$$

Now, if we have $y_n^v + [0, h_n] \cdot f(B) \subseteq B$, then for every $u \in S', Tu \in S'$, so that S' is mapped into itself by T. Thus there is a unique solution to the ODE on $[x_n, x_{n+1}]$ that has values in B. We then just need to get a box $[\tilde{y_n}]$ such that $y_n^v + [0, h_n] f([\tilde{y_n}]) \subseteq [\tilde{y_n}]$. Let P be the interval Picard-Lindelöf operator defined by $P(R) = [y_n] + [0, h] f(R)$. We find $[\tilde{y_n}]$ by iterating P: we start from R_0 which contains $[y_n]$ and y_{n+1}^*, and we compute $R_n = P(R_{n-1})$. We stop once we found R_m such that $R_{m+1} \subseteq R_m$. We use R_{m+1} as our a priori enclosure for the values of $y(x; x_n, y_n^v)$, and then compute the over-approximation:

$$\frac{d^4 f}{dx^4}\big(y(\xi; x_n, y_n^v)\big) \in \frac{d^4 f}{dx^4}\big(R_{m+1}\big). \tag{3.2}$$

So, combining 3.1 and 3.2 gives an enclosure of η_{n+1}:

$$\eta_{n+1} \in \frac{d^4 f}{dx^4}\big(R_{m+1}\big) - \frac{d^5(\phi_n)}{dx^5}\big([x_n, x_{n+1}]\big) \tag{3.3}$$

3.2. Propagation of the Error

The propagation of the error at the nth step $[\epsilon_n]$ into the error at the $(n+1)$st step must account for the separation between the solutions of Equation (1.1) with an initial value y_n^v at x_n and with an initial value y_n: the point y_{n+1}^* is the application ψ_n at y_n^v, and y_{n+1}^r is the application of ψ_n at y_n. These two flows are functions defined over $D \subseteq \mathbb{R}^d$ with values in $D' \subseteq \mathbb{R}^d$. The separation of such functions is computed using the Jacobian matrix. For every differentiable function $f : D \to \mathbb{R}^d$, let

$$J(f) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_d} \\ & \cdots & & \\ \frac{\partial f_d}{\partial x_1} & \frac{\partial f_d}{\partial x_2} & \cdots & \frac{\partial f_d}{\partial x_d} \end{pmatrix}.$$

be its Jacobian matrix, and let

$$J(f, Y) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(Y) & \frac{\partial f_1}{\partial x_2}(Y) & \cdots & \frac{\partial f_1}{\partial x_d}(Y) \\ \frac{\partial f_2}{\partial x_1}(Y) & \frac{\partial f_2}{\partial x_2}(Y) & \cdots & \frac{\partial f_2}{\partial x_d}(Y) \\ & \cdots & & \\ \frac{\partial f_d}{\partial x_1}(Y) & \frac{\partial f_d}{\partial x_2}(Y) & \cdots & \frac{\partial f_d}{\partial x_d}(Y) \end{pmatrix}.$$

be its Jacobian matrix with all derivatives evaluated at some $Y \in D$. We may now express the extension of the mean value theorem to multivariate functions in a compact formulation:

Theorem 3.1 *Let $D \subseteq \mathbb{R}^d$ be an open subset of $\mathbb{R}^d$ and let $f : D \to \mathbb{R}^d$ be a differentiable function on D. Then, for all a and u such that $[a, a+u] \subset D$, there exists $\theta \in]0, 1[$ such that*

$$f(a+u) - f(a) = J(f, a + \theta \cdot u) \cdot u.$$

Now, as $y_{n+1}^* = \psi_n\big(y_n^v\big)$ and $y_{n+1}^r = \psi_n\big(y_n\big)$, there must exist $\theta \in]0, 1[$ and $c = y_n^v + \theta(y_n - y_n^v)$ such that:

$$y_{n+1}^* - y_{n+1}^r = J\big(\psi_n, y_n^v + c\big) \cdot \big(y_n - y_n^v\big). \tag{3.4}$$

This formula gives the exact value of χ_{n+1}, assuming we know the exact value of c and y_n^v, which we do not. However, the following holds:

$$\forall \theta \in]0, 1[, y_n^v + \theta\big(y_n + y(x_n)\big) \in [y_n, y(x_n)] \subseteq y_n + [\epsilon_n].$$

Furthermore, we know that $y_n^v \in y_n + [\epsilon_n]$, so we may enclose the propagation of $[\epsilon_n]$ as follows:

$$\begin{aligned}\chi_{n+1} &= J\big(\psi_n, y_n^v + c\big) \cdot \big(y_n - y_n^v\big) \\ &\in J\big(\psi_n, y_n + [\epsilon_n]\big) \cdot [\epsilon_n].\end{aligned} \tag{3.5}$$

Equation (3.5) gives a method for computing χ_{n+1}. However, if we apply this formula naively, the wrapping effect mentioned in Section 1.2 makes the error grow. Actually, if the Jacobian matrix is a rotation matrix, then huge overestimations are performed at each step, as shown by Figure 2 (where $J = J\big(\psi_n, y_n + [\epsilon_n]\big)$). To limit this problem, we can use the same techniques as Taylor Series methods, for example Löhner's QR factorization method [10]. Its idea is the following: instead of representing errors as boxes in the standard orthogonal coordinate system, we will express them in a different, better basis before performing the multiplication with the Jacobian matrix. The new coordinate system is constructed as follows: the first axis is chosen parallel to the longest edge of J, and we construct the other axis so that they are as parallel as possible to other edges. This is achieved by permuting the columns of J and then computing its QR-factorization. For more details, see [12].

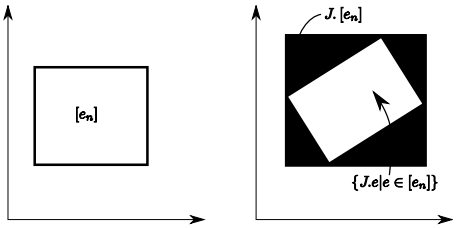


Figure 2. Wrapping effect.

$$a = f_a + e_a\overrightarrow{\varepsilon_e} \text{ and } b = f_b + e_b\overrightarrow{\varepsilon_e}$$

$$\begin{array}{rcl} a+b & = & \uparrow_\sim (f_a+f_b) + (e_a+e_b + \downarrow_\sim (f_a+f_b))\overrightarrow{\varepsilon_e} \\ a-b & = & \uparrow_\sim (f_a-f_b) + (e_a-e_b + \downarrow_\sim (f_a-f_b))\overrightarrow{\varepsilon_e} \\ a\times b & = & \uparrow_\sim (f_a \times f_b) \\ & & \quad + \big(e_a f_b + e_b f_a + e_a e_b + \downarrow_\sim (f_a \times f_b)\big)\overrightarrow{\varepsilon_e} \end{array}$$

Table 1. Global error arithmetic.

3.3. Roundoff Errors

Roundoff errors occur during the computation of the next approximation point y_{n+1}. They are due to the difference between the computation made on a finite precision machine, which leads to the floating point number y_{n+1}, and the computation that would be performed on real numbers, which leads to the real value y^r_{n+1}. This separation comes from the implementation of floating point numbers as defined by the IEEE754 Standard. They may be reduced using multi-precision arithmetic, such as the MPFR library [5], but not eliminated and we thus need to consider very carefully these error terms. We compute together with the floating point number y_{n+1} an interval $[e_{n+1}]$ such that $y^r_{n+1} \in y_{n+1} + [e_{n+1}]$. This interval which overapproximates all computation errors is computed using the *global error* arithmetic, first defined and used in the field of numerical validation of C programs [14]. The idea is to attach to each floating point number a formal term which represents the distance between the floating point and the real number it is supposed to represent. Thus, a global error number a may be written as:

$$a = f_a + [e_a]\overrightarrow{\varepsilon_e},$$

where f_a is a floating point number and $\overrightarrow{\varepsilon_e}$ is a formal variable. $[e_a]$ is supposed to be the difference between the real value x_a and its floating point representation f_a. It should thus be a real number, and we naturally implement it as an interval. This representation means that a is a floating point number with value f_a which represents a real number x_a such that $x_a \in f_a + [e_a]$. We have two kinds of information: the result of a floating point computation and its distance to the real value. The main advantage of this representation is that the final enclosure on x_a, namely $f_a + [e_a]$, does not need to contain f_a, whereas all interval based representations (either infimum/supremum or midpoint/radius) produce enclosures that contain both the real value and the floating point number. Thus, the width of our error term $[e_a]$ is usually smaller than the width of the otherwise computed interval. Moreover, it is possible to use more precision for the computation of the error term than for the floating point term, in order to decrease the width of $[e_a]$. In our implementation, we use double the precision for the error terms.

Let us give an example to explain the gain of this arithmetic compared to interval arithmetic. Suppose that we have floating point numbers with a mantissa of 5 digits only. We start from $y = 1$ and subtract 10^6 times 10^{-6} to y. In real number arithmetic, the result is obviously 0. In floating point arithmetic, the result will be $y = 1$ as a cancellation occurs at each subtraction. If we do the same computation with the infimum/supremum interval arithmetic, we obtain a final interval of width 10^{66}. With the midpoint/radius arithmetic, using doubled precision for the radius terms, we obtain $(1, 1.0001)$, i.e. an enclosure of width 2 for the real result. With the global error arithmetic, we obtain $1 + [-1.0001, -9.9977]\overrightarrow{\varepsilon_e}$, i.e. an enclosure of width $4 \cdot 10^{-4}$. This difference[1] is due to the fact that the error term in the global error representation is *directed*: it is not a radius indicating in which circle the real value lies but rather an arrow aiming at it. For a more complete comparison between various arithmetic used for the validation of numerical programs, see [11].

The casting of a real number x into a global error number a proceeds as follows: the floating point part of a is the closest floating point number to x, denoted $\uparrow_\sim (x)$, and its error part is the distance between x and $\uparrow_\sim (x)$, denoted $\downarrow_\sim (x)$. Thus we have:

$$a = \uparrow_\sim (x) + \downarrow_\sim (x)\overrightarrow{\varepsilon_e}.$$

In a computer, the value $\downarrow_\sim (x)$ is enclosed by the interval $[-u, u]$, where u is the value of the last bit of the representation of $\uparrow_\sim (x)$, and doubled precision is used. Basic operations and elementary functions are defined over such numbers. The rules for addition, subtraction and multiplication are given in Table 1. So, if we use this arithmetic to compute y_{n+1}, we not only get the floating point value of the next step but also its distance to the real value y^r_{n+1}:

$$\begin{array}{rcll} y^r_{n+1} & = & y_{n+1} + [e_{n+1}]\overrightarrow{\varepsilon_e} & \\ y^r_{n+1} - y_{n+1} & \in & [e_{n+1}]. & (3.6) \end{array}$$

[1] A c++ program showing these results can be downloaded at `http://www.lix.polytechnique.fr/Labo/Olivier.Bouissou/progs/patriot.cc`

3.4. Putting Things Together

Using formulas (3.3), (3.4) and (3.6), we have:

$$\begin{aligned}\epsilon_{n+1} &= \eta_{n+1} + \chi_{n+1} + e_{n+1} \\ &\in \frac{d^4 f}{dx^4}(R) - \frac{d^5(\phi_n)}{dx^5}([x_n, x_{n+1}]) + \\ &\quad J(\psi_n, y_n + [\epsilon_n]).[\epsilon_n] + [e_{n+1}]. \qquad (3.7)\end{aligned}$$

4. Controlling the Step Size

The global error can be estimated with our method and interval arithmetic. Using this error estimator, one may want to control the error by modifying the step-size in order to achieve a prescribed accuracy. The main difficulty is that it is in general very expensive to control the global error. Actually, as it has been shown, the global error after $n+1$ steps is computed as:

$$y_{n+1} - y(x_{n+1}) = \big(y_{n+1} - y^*_{n+1}\big) + \big(y^*_{n+1} - y(x_{n+1})\big).$$

The second term represents the local error due to the method and its implementation, while the first one represents the stability of the dynamical system (in the sense of Lyapunov's stability [16]): given two close initial points, a stable system will join them whereas an unstable system will separate them. The second term depends on the chosen method, and we can control it, whereas the first term depends on the problem itself, and it is therefore not directly observable. If the problem is unstable between x_n and x_{n+1}, this term grows, and a reduction in the step-size does not affect this. Moreover, if we want to achieve a certain tolerance at time T, it may not be efficient to control the global error before this point, as the integral curves may converge only near T. So, controlling the global error generally requires at least two integrations of the problem, as the step-size selection at one point strongly depends on what happens next. Here we concentrate on the local error control for performance issues. Another reason why we should do that comes from the previous formula. Clearly, keeping the local error (the second term) small will affect the global error and help to keep it small. Moreover, control techniques often try to prevent from instability phenomenas and thus keep the values of y_n inside the stability region of the problem: the control techniques we present will reduce the step-size as soon as the derivative of the solution grows. Thus, the accuracy of the computation will be increased in the regions where the derivative is high, so that the first term of the formula will be kept low.

For all these reasons, we focus on the following problem: given a user-defined, absolute tolerance tol, adapt the step-size so that the error introduced at each step (both method and roundoff error) is smaller than tol. Obviously, as the step-size decreases, the method error decreases. However, the dependence of the roundoff error into the step-size is not so obvious, and this error clearly limits the accuracy one can obtain. Therefore, we only control the method error by adapting the step-size. If one wants to control the roundoff error as well, then increasing the precision of the computation by using multi-precision arithmetic such as the MPFR library would be the easiest way.

The method for controlling the step-size is derived from general methods of control theory for the automatic control of physical systems. The main idea is that the step-size is the adjustment variable to control the truncation error, which can be seen as a physical variable with an optimal value, tol, and that we must bring as close as possible to it. The dependence between the error and the step-size is given by the formula

$$\epsilon_{n+1} = |\varphi_n| \cdot h_n^5,$$

where $|\varphi_n|$ is the enclosure of the value of the fifth derivative of f on the a priori enclosure for the nth step (between x_n and x_{n+1}). This gives us a first control method, based on the assumption that $|\varphi_n| \approx |\varphi_{n+1}|$:

$$h_{n+1} = \left(\frac{tol}{|\varphi_{n+1}| h_n^5}\right)^{\frac{1}{5}} h_n$$

This controller is called the *integral* controller. Actually, if you use this formula, then the error after the $n+1$st step is $\epsilon_{n+2} = |\varphi_{n+2}|.h_{n+1}^5 = |\varphi_{n+1}|.\frac{tol}{|\varphi_{n+1}|.h_n^5}.h_n^5 = tol$. However, the assumption that $|\varphi_n| \approx |\varphi_{n+1}|$ is in general false for non trivial problems. Hence, this control mechanism tends to overcompensate, leading to many variations in the step-size and to rejected steps. To overcome this problem, a first solution would be to use $\theta.tol$ instead of tol in the formula, with $\theta \in [0,1]$. This smooths slightly the variations of h_n, but this strategy is still not satisfactory for complex problems. Hence, we build a more complex controller which considers not only the previous step error to compute the next step-size, but also takes into account the variation of the error over the last two steps. In this way, if the error is growing, then the previous controller will be adjusted so that the step-size does not increase too much. On the contrary, if the error is decreasing, the controller will amplify the variations of the step-size. The idea of this second controller is to add, as in control theory, a term proportional to the control error to the previous integral term. This leads to the formula:

$$h_{n+1} = \left(\frac{\theta.tol}{r_{n+1}}\right)^{k_1} \left(\frac{r_n}{r_{n+1}}\right)^{k_p} h_n,$$

with $r_n = |\varphi_{n+1}|.h_n^5$. k_1 and k_p are the controller parameters. The difficulty is to chose them to achieve a good performance. Ideally, they should be computed independently for every problem, as they strongly depend on its condition. However, choosing $k_1 = \frac{0.3}{k}$ and $k_p = \frac{0.2}{k}$ is a good choice for many problems.

5. Numerical Results and Benchmarks

We implemented this method in a C++ library that offers an easy-to-use framework for solving differential equations. Our implementation depends on two libraries:

- GiNaC [1], a formal derivation C++ library. With this tool, we generate at compile time C++ code for the higher order derivatives. We thus provide a source code transformation system which computes all the function needed for the computation of the error term.
- Profil/BIAS [9], a C++ interval library.

In this section, we study the performances of our library. First, we compare the speed and accuracy of our library with Taylor series methods; we chose to compare with AWA [10] and VNODELP [13]. AWA was the first software to use Taylor series to achieve guaranteed integration, and VNODELP is the new version of VNODE, which is probably the validated solver that is used the most. We run the tests with GRKLib, VNODELP and AWA but also with VNODELP and AWA limited to order 5. Actually, the main limitation of our method is its relatively low order, but our RK4 method is much more efficient than Taylor series method of the same order, while as precise as them. We then show how the method scales with problem size, tolerance and initial error. The tests were run on a laptop (two 1.7Ghz processors, 1Gb of RAM) running Ubuntu Linux. We used g++ (version 4.1) with optimization options -O2.

5.1. Linear problem: pure contraction

We integrate the IVP (5.1) on the interval $[0, 2000]$, and with an absolute tolerance of 10^{-12} at each step. This problem is a pure contraction, meaning that the Jacobian matrix does not rotate the error terms. Thus, the wrapping effect described in 3.2 is very limited.

$$\dot{Y} = \begin{pmatrix} -0.4375 & 0.0625 & 0.2652 \\ 0.0625 & 0.4375 & 0.2652 \\ -0.2652 & 0.2652 & 0.375 \end{pmatrix} Y \quad Y_0 = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \end{pmatrix} \tag{5.1}$$

Figure 3 shows how the CPU time depends on the integration time. VNODELP is 1.5 times faster than our library, but we are 3 times faster than VNODELP with order 5. AWA is more than 10 times slower than VNODELP of order 5, whatever the order is. The time for one step is much smaller for GRKLib ($4.6 \cdot 10^-5$ seconds against $3.8 \cdot 10^-4$ for VNODELP). The width of the final enclosure is $2 \cdot 10^{-9}$ for our method, 3.10^{-10} for VNODELP with order 5 and 10^{-12} for VNODELP and AWA. The error is bigger for GRKLib because the error control method controls the step size so that the local error introduced is of 10^{-12}, and we do not control the global error. If we integrate the same problem with fixed, small step size, we can achieve a final global error of 10^{-14} while being only 5 times slower. Thus, we would greatly benefit from a global error control which would predict the evolution of the step size in a better way.

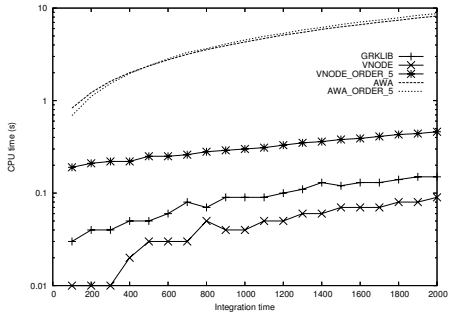


Figure 3. CPU time versus t for Problem (5.1)

5.2. Linear problem: pure rotation

We solve the IVP (5.2), on the interval $[0, 2000]$, with an absolute tolerance of 10^{-12} at each step. This problem is a pure rotation, meaning that the Jacobian matrix is almost a rotation matrix. Thus, it widely suffers from wrapping effect, and this problem shows the efficiency of Löhner's QR factorization method.

$$\dot{Y} = \begin{pmatrix} 0 & -0.707107 & 0.5 \\ 0.707107 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix} Y \quad Y_0 = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \end{pmatrix} \tag{5.2}$$

Figure 4 shows how the CPU time depends on the integration time. The time per step remains 10 times smaller for GRKLib, but we need to make more steps because of our limited order. The width of the error at $t = 2000$ is 2.10^{-9} for GRKLib, 10^{-11} for VNODELP and AWA and 6.10^{-10} for VNODELP with order 5. Once again, a global error control method would help to reduce the error width.

5.3. Non linear problem: Lorenz equations

Finally, we applied our method on Lorenz equations (5.3). We set the absolute tolerance to 10^{-12} and performed the integration on the interval $[0, 15]$.

$$\begin{cases} \dot{y_1} = \sigma.(y_2 - y_1) \\ \dot{y_2} = y_1.(\rho - y_3) - y_2 \\ \dot{y_3} = y_1.y_2 - \beta.y_3 \end{cases} \quad \begin{cases} y_1(x_0) = 15.0 \\ y_2(x_0) = 15.0 \\ y_3(x_0) = 36.0 \end{cases} \tag{5.3}$$

Figure 5 shows how the computation time depends on the time. Once again, the time per step is much smaller for GRKLib, but as the derivatives of the function take very high values, we need to make very small steps to achieve

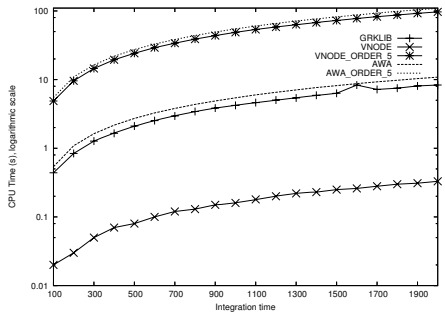


Figure 4. CPU time versus t for Problem (5.2)

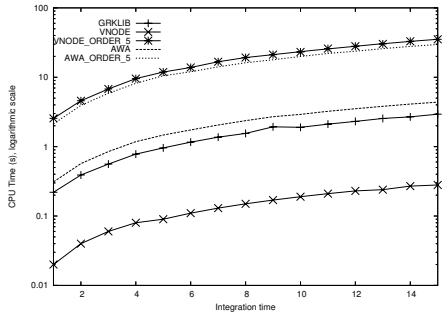


Figure 5. CPU time versus t for Problem (5.3)

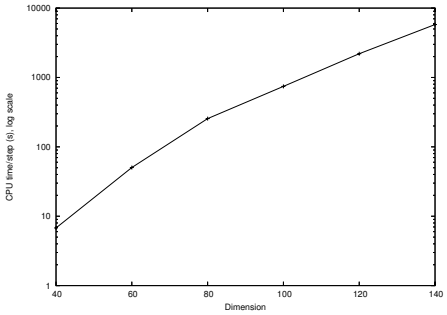


Figure 6. CPU time per step versus n

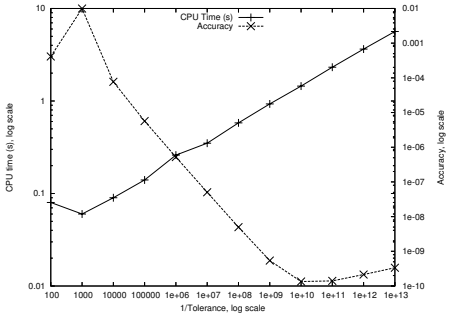


Figure 7. CPU time and accuracy versus tol

the given tolerance. However, we remain 10 times faster than VNODELP with order 5 and 2 times faster than AWA. The final error at $t = 15$ is $4 \cdot 10^{-3}$ for GRKLib, $5 \cdot 10^{-6}$ for VNODELP, $2 \cdot 10^{-3}$ for VNODELP with order 5 and 10^{-4} for AWA. AWA with order 5 did not make it to $t = 15$. The bigger number of steps we have to make explains this difference.

5.4. Performance

Work versus problem size

To study how computation time depends on the problem dimension, we used the DETEST problem C3 [7], as suggested by Nedialkov [13]. The problem is given by:

$$\dot{Y} = \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ & & & \vdots & & \\ 0 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix} Y$$

with $y_0 = (1, 0, \dots, 0)^T$. We solved the equation from 0 to 2 for dimensions $n = 40, 60, 80, \dots, 140$. Figure 7 shows the computation time per step (we need 42 steps to reach 2). The complexity is $O(n^3)$ because the QR method to reduce the wrapping effect requires a matrix decomposition which is the most time expensive part of the algorithm.

Work versus tolerance

To study the dependence of our method on the tolerance, we integrate the linear problem (5.1) on the interval $[0, 1000]$ with a tolerance $tol = 5.10^{-2}, 10^{-2}, 5.10^{-3}, \dots, 5.10^{-14}$. Figure 7 shows how CPU time and accuracy depends on the tolerance. As excepted, the accuracy increases as the tolerance decrease, whereas the CPU time increases.

Accuracy versus initial conditions

To study how the method depends on initial conditions, we integrate the Lorenz equations (5.3) with initial errors $tol = 10^{-4}, 10^{-5}, \dots, 10^{-9}$. We present on Figure 5.4 the maximum time at which we were able to integrate (5.3) for each value of tol. As excepted, the bounds diverge much quicker than for point initial conditions; on this problem, the bounds tend to explode as soon as they start diverging.

6. Conclusion

In this article, we have shown the feasibility of a guaranteed version of the classical numerical algorithm RK4. The guaranteed nature of our method differs from previous

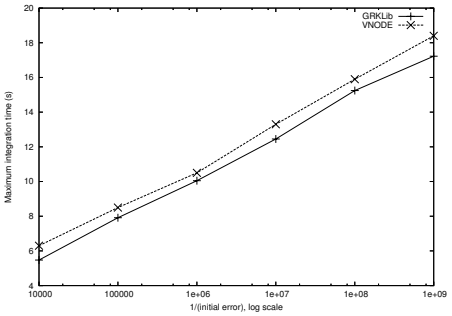


Figure 8. Maximum T_{end} versus initial error

works in the sense that we compute non validated points in a first step and then guaranteed error bounds. To compute these bounds, we localize every type of errors that may occur during the computation of the approximating points: truncation errors due to the method, propagation errors due to the dynamical system, and roundoff errors due to the precision of machines on which the method is implemented. We use a constant a priori enclosure to compute truncation errors, Löhner's method to reduce the wrapping effect in propagation errors, and the global error domain for roundoff errors. We also use a step-size control mechanism that gives good performance results.

We believe that this method is promising because it does not mix interval and floating point computations. Actually, intervals are used only for verification tasks and not for the computation of the next approximation points. This clear separation is analogous to the separation between floating point numbers and intervals in global error arithmetic. This has given very good results in the field of numerical validation, so we are confident that it should perform well for guaranteed integration.

The idea of separation between interval and floating point computations is orthogonal to the choice of the numerical method. Actually, all we need is an iteration function for which the derivatives are computable. We chose to base our library first on Runge-Kutta methods because they are widely used for numerical (non guaranteed) integration and that users generally know how to tune the step size control mechanism to make the method as precise as possible. However, we do not want to limit ourselves to this particular integration scheme, and we plan to add more sophisticated ones to our library to fit better to more types of problems. Especially, we are planning to use higher order numerical integration schemes; experimentations showed that the mean step size is the main limitation to performance and accuracy, the use of higher order methods will allow bigger step sizes and consequently much better performances.

References

[1] C. Bauer, A. Frink, and R. Kreckel. Introduction to the GiNaC framework for symbolic computation within the C++ programming language. *Journal of Symbolic Computation*, 33(1):1–12, 2002.

[2] L. Bieberbach. On the remainder of the runge-kutta formula in the theory of ordinary differential equation. *ZAMP*, 2:233–248, 1951.

[3] J. C. Butcher. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, New York, 1987.

[4] G. Corliss and Y. F. Chang. Solving ordinary differential equations using Taylor series. *ACM Transaction on Mathemathical Software*, 8(2):114–144, 1982.

[5] G. Hanrot, V. Lefevre, R. F., and P. Zimmermann. The MPFR library. Available at `www.mpfr.org`.

[6] T. Henzinger and P. Ho. Algorithmic analysis of nonlinear hybrid systems. In *CAV*, volume 939 of *LNCS*, pages 225–238. Springer Verlag, 1995.

[7] T. Hull, W. Enright, B. Fellen, and A. Sedgwick. Comparing numerical methods for ordinary differential equations. *Journal on Numerical Analysis*, 9(4):603–637, 1972.

[8] M. Kieffer and E. Walter. Guaranteed nonlinear state estimator for cooperative systems. *Journal of Numerical Algorithms*, 37(1–4):187–198, Dec. 2004.

[9] O. Knüppel. PROFIL/BIAS – A fast interval library. *Computing*, 53:277–287, 1994.

[10] R. Löhner. *Einschliessung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen*. PhD thesis, Universität Karlsruhe, 1988.

[11] M. Martel. An overview of semantics for the validation of numerical programs. In *VMCAI*, volume 3385 of *LNCS*, pages 59–77. Springer, 2005.

[12] N. Nedialkov, K. Jackson, and G. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[13] N. S. Nedialkov. Interval tools for ODEs and DAEs. Technical Report CAS 06-09-NN, Dept. of Computing and Software, McMaster University, 2006.

[14] S. Putot, E. Goubault, and M. Martel. Static analysis-based validation of floating-point computations. In *Numerical Software with Result Verification*, volume 2991 of *LNCS*, pages 306–313. Springer, 2003.

[15] A. Rauh, E. Auer, E. Hofer, and W. Luther. Validated modeling of mechanical systems with SmarMOBILE: Improvement of performance by ValEncIA-IVP. In *Reliable Implementation of Real Number Algorithms: Theory and Practice*. Springer Verlag, to appear.

[16] S. Sastry. *Nonlinear Systems Analysis, Stability and Control*. Spinger, Berlin, 1999.

[17] O. Stauning. *Automatic Validation of Numerical Solutions*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 1997.

[18] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer Verlag, New York and Berlin, 1993.

Hardware Implementation of Continued Logarithm Arithmetic

Tomáš Brabec
Faculty of Electrical Eng. at Czech Technical University
Dept. of Computer Science and Engineering
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
brabect1@fel.cvut.cz

Abstract

This paper gives details on architecture of an arithmetic unit built on principles of continued logarithms and provides its sample characterization using FPGA technology. Continued logarithms can be used for exact arithmetic, but similarly to other exact/reliable methods they face the instant problem of poor performance. Their naturally binary character, however, gives them a good potential to be realized directly in hardware. We prove feasibility of this approach by constructing a continued logarithm unit and quantifying its possible performance.

1. Introduction

Continued logarithms form a number system capable of representing real numbers. As it relies on error-free rational arithmetic, it is useful for exact arithmetic, which in other words means that computing with continued logarithms produces accurate results with arbitrary precision [7]. This paradigm was first introduced in [5] as a derivative of regular continued fractions and has a better potential for binary arithmetic and hardware implementation. Likewise continued fractions, continued logarithms feature many advantageous properties such as reversibility and incrementality of computation, implicit error bound tracking, good rational approximation and etc. [6].

Unfortunately, continued logarithms also keep the major problem of continued fraction arithmetic, that is the high complexity of arithmetic algorithms. These algorithms generally compute bilinear fractional transformations operating with eight-tuples of big-integer coefficients [5, 9]. This introduces significant computational overhead, which is especially notable for sequential software implementation and which can practically prevail over most advantages of continued representations.

There is, however, a chance to minimize this overhead by exploiting parallelism inside the (bi)linear fractional transformations and use dedicated hardware support. This possibility is further supported by regularity and uniformity of arithmetic algorithms, but its final success relies on efficient use of hardware resources. Even though continued fractions in their regular form are not generally useful in this sense, there were some past attempts for their hardware implementation [9] using a factorization into binary digits. More recently a similar idea appeared in [10].

Our intention here is to investigate the use of continued logarithms as a possible alternative and show its promise for practical realization. For this reason, we cover the most important aspects of continued logarithms in Section 2 – the representation, principles of arithmetic, and some important properties. We also compare continued logarithms with regular continued fractions, showing on some practical experiments that there is no major difference between them. We try to keep that section compact but still complete so as to provide sufficient material for understanding the remaining parts.

The idea of constructing a continued logarithm unit was already introduced in [2], but its description was rather general and we showed only preliminary results. Here we give a more detailed and complete overview of the unit's architecture (Sections 3.1 and 3.2) and present final results of its implementation (Section 3.3). We also provide an analysis in Section 3.4 that quantifies a performance potential of the unit, showing a significant improvement over the software implementation.

There is, however, a limitation behind the continued logarithm representation, which affects applicability of the proposed unit. This problem and its possible solution is finally discussed in Section 4.

2. Continued Logarithm Arithmetic

2.1. Basics

Continued logarithm representation of a real number x is defined by a recursive formula (1), which produces one

Table 1. BCL "Cubic" transformations of fractional coefficients during processing/generating a digit of individual variables.

Digit	x-transformation	y-transformation	z-transformation
–	$\begin{pmatrix} -A & B & -C & D \\ -E & F & -G & H \end{pmatrix}$	$\begin{pmatrix} -A & -B & C & D \\ -E & -F & G & H \end{pmatrix}$	$\begin{pmatrix} -A & -B & -C & -D \\ E & F & G & H \end{pmatrix}$
/	$\begin{pmatrix} B & A & D & C \\ F & E & H & G \end{pmatrix}$	$\begin{pmatrix} C & D & A & B \\ G & H & E & F \end{pmatrix}$	$\begin{pmatrix} E & F & G & H \\ A & B & C & D \end{pmatrix}$
0	$\begin{pmatrix} A+B & A & C+D & C \\ E+F & E & G+H & G \end{pmatrix}$	$\begin{pmatrix} A+C & B+D & A & B \\ E+G & F+H & E & F \end{pmatrix}$	$\begin{pmatrix} E & F & G & H \\ A-E & B-F & C-G & D-H \end{pmatrix}$
1	$\begin{pmatrix} 2A & B & 2C & D \\ 2E & F & 2G & H \end{pmatrix}$	$\begin{pmatrix} 2A & 2B & C & D \\ 2E & 2F & G & H \end{pmatrix}$	$\begin{pmatrix} A & B & C & D \\ 2E & 2F & 2G & 2H \end{pmatrix}$

digit per iteration starting from the most significant one. As (1) consists of four definition cases, there are in total four digits ('$-$', '/', '0' and '1') respectively assigned to individual cases. Thus starting from the top, digit '$-$' corresponds to $-x_i$, '/' to $1/x_i$ and etc. It happens from the nature of (1) that the digits '$-$' and '/' have "sign" character as they may appear only at the beginning of a number representation. This is why we denote this representation a *binary continued logarithm* (BCL), even though there are actually four digits.

$$x_{i+1} = \begin{cases} -x_i & \text{for } x_i \in [-\infty, 0), \\ 1/x_i & \text{for } x_i \in [0, 1), \\ 1/(x_i - 1) & \text{for } x_i \in [1, 2), \\ x_i/2 & \text{for } x_i \in [2, +\infty], \end{cases} \tag{1}$$
$$i = 0, 1, \ldots \text{ and } x_0 = x \in \mathbb{R}.$$

Let us show the use of (1) to find a BCL representation of a rational number $x = -\frac{2}{3}$. We start from $i = 0$ by letting $x_0 = x$ and iteratively continue to higher indices as shown below, finding out that $-\frac{2}{3} = (-/010)_{\text{BCL}}$.

i	x_i	digit	condition	x_{i+1}
0	$-\frac{2}{3}$	'–'	$[-\infty, 0)$	$-(-\frac{2}{3})$
1	$\frac{2}{3}$	'/'	$[0, 1)$	$1/\frac{2}{3}$
2	$\frac{3}{2}$	'0'	$[1, 2)$	$1/(\frac{3}{2} - 1)$
3	2	'1'	$[2, +\infty]$	$2/2$
4	1	'0'	$[1, 2)$	$1/(1 - 1)$
5	∞			

Although it is not explicitly defined, the recursive process may terminate once the value of x_i reaches ∞. By terminating the digit sequence for such i, we avoid an infinite suffix of trailing ones caused by equivalence $\frac{\infty}{2} = \infty$.

Recursive nature of (1) and the employed rational functions imply that the BCL representation has a character of composed linear fractional transformations (LFTs). This character is typical for many similar representations (e.g. [6, 10, 11]), all of which base their arithmetic on a bilinear fractional transformation (BLFT). BLFT is a rational function of the following form

$$\begin{aligned} z(x, y) &= \frac{Axy + By + Cx + D}{Exy + Fy + Gx + H} \\ &= \begin{pmatrix} A & B & C & D \\ E & F & G & H \end{pmatrix} (x, y), \end{aligned} \tag{2}$$

where A through H are integer coefficients and x, y are real-valued arguments, in our case having a form of continued logarithms.

Notice that the BLFT function is general enough to compute all basic arithmetic operations, being the only sufficient operator to constitute whole continued logarithm arithmetic. As explained in [4], BLFT is closed to composition of LFTs and so it naturally matches the character of continued logarithms. This closure actually means that the bilinear form of BLFT stays preserved, no matter what input (x, y) and output (z) digits are being processed.

It thus follows that after processing k digits from x and l from y and after generating i digits of a result z, the state of (2) changes from $z(x, y) = z_0(x_0, y_0)$ to $z_i(x_k, y_l)$, defined by

$$z_i(x_k, y_l) = \begin{pmatrix} A_i^{k,l} & B_i^{k,l} & C_i^{k,l} & D_i^{k,l} \\ E_i^{k,l} & F_i^{k,l} & G_i^{k,l} & E_i^{k,l} \end{pmatrix} (x_k, y_l). \tag{3}$$

This new state is solely characterized by actual values of fractional coefficients and indices i, k, l. Notice that each fractional coefficient has associated all three indices, as its value changes with processing of every digit, input or output. The way these coefficients change depends on a variable and a digit being processed.

The four digits and the three variables account together to twelve possible transformations, which are summarized in Tab. 1. The table shows a new state of a BLFT (2) transformed correspondingly to a given variable and a processed

digit. One can derive these transforms easily [2] by substituting for a given variable from the BCL definition formula (1) into (2). A partial example of their use is shown below, but the particular details are left to Section 3.1.

The last substantial aspect of the continued logarithm arithmetic is its understanding as a second-level arithmetic with the underlying variable-length integer arithmetic being the first level. It means that, despite of x, y, z having BCL representation, the coefficients A to H of (2) are integers represented in a normal, positional number system. Therefore, all operations with these coefficients found in Tab. 1 use conventional integer arithmetic. This understanding is very important and we will return to it in the following sections, where we inspect properties of continued logarithms.

Example: We illustrate the continued logarithm arithmetic on computing a function $z = xy$, where we put for simplicity $x = y = 2 = (10)_{\mathrm{BCL}}$. We use here a special notation where the symbol $\overset{d}{=}_v$ denotes an equivalence attained by processing a digit d from an input variable v. Likewise, the symbol $\overset{d}{\rightarrow}_z$ represents an output z-transformation, during which an output digit d is emitted.

$$z_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} (2,2) \overset{\text{'1'}}{=}_x \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} (1,2)$$
$$\overset{\text{'1'}}{=}_y \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} (1,1) \geq 4 \quad \overset{\text{'1'}}{\rightarrow}_z z_1$$
$$z_1 = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} (1,1) \geq 2 \quad \overset{\text{'1'}}{\rightarrow}_z z_2$$
$$z_2 = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} (1,1) \overset{\text{'0'}}{=}_x \begin{pmatrix} 4 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \end{pmatrix} (\infty,1)$$
$$\overset{\text{'0'}}{=}_y \begin{pmatrix} 4 & 4 & 4 & 4 \\ 4 & 0 & 0 & 0 \end{pmatrix} (\infty,\infty) = 1 \quad \overset{\text{'0'}}{\rightarrow}_z z_3$$
$$z_3 = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 4 & 4 \end{pmatrix} (\infty,\infty) = \infty$$

We can see that the produced sequence '1', '1', '0' corresponds to a BCL representation of a number $z = 4$.

2.2. Properties

The BCL definition formula does not directly evoke that continued logarithms are really continued fractions, but this fact is immediate from formulas (4) and (5) introducing a *canonical continued logarithm* (CCL). Both BCL and CCL representations are equal and they are tied together through powers of two so that every k_i within CCL represents the number of '1' digits between i-th and $(i-1)$-th '0' digit within BCL (see [5] for details).

$$x = s\,(\chi)^e\,, \tag{4}$$

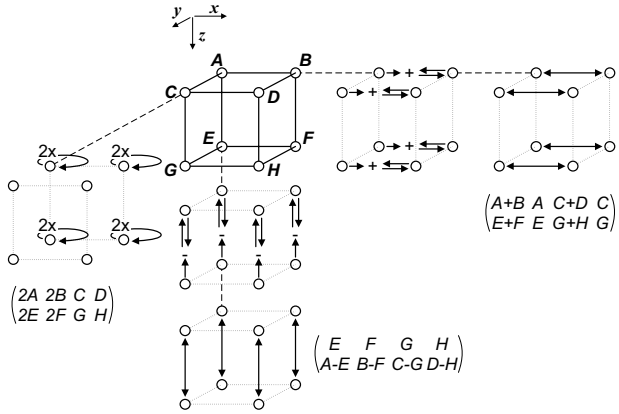


Figure 1. Graphical view of some BCL cubic transformations (see Tab. 1) after mapping fractional coefficients onto a cube.

where s, e and χ are defined as follows:

$$s = \begin{cases} -1 & \text{for } x < 0, \\ +1 & \text{otherwise,} \end{cases}$$
$$e = \begin{cases} -1 & \text{for } |x| < 1, \\ +1 & \text{otherwise,} \end{cases}$$
$$\chi = 2^{k_0} + \cfrac{2^{k_0}}{2^{k_1} + \cfrac{2^{k_1}}{\ddots + \cfrac{2^{k_{n-1}}}{2^{k_n} + \ldots}}} \geq 1. \tag{5}$$

The canonical form is especially useful for studying theoretical properties of continued logarithms, because it is similar to the form of regular continued fractions. The difference is that regular continued fractions use a "linear" approximation of a real number x by an integer $\lfloor x \rfloor$, while continued logarithms use a "logarithmic" approximation by a close power of two, namely $2^{\lfloor \log_2 x \rfloor}$. Similarity of this principle suggests analogy in properties of continued logarithms and regular continued fractions. Let us mention for instance uniqueness of representation, built-in error analysis, reversibility of computation, etc. [6]. Proofs of these properties known for regular continued fractions [8] can be easily adapted to continued logarithms. The logarithmic nature, on the other hand, makes some properties distinct. A particular example is approximation convergence, which is much slower for continued logarithms.

Other interesting properties are due to theoretical principles of continued arithmetic and they are thus common to both considered continued representations. It is especially the uniformity and regularity of BLFT transformations, which is nicely illustrated by mapping the fractional coefficients of (2) onto a cube (see Fig. 1). Individual cube

dimensions correspond to the three variables of a bilinear transformation and at the same time they define directions, in which coefficient transformations happen. For details refer to [9]. The graphical view also reveals high degree of parallelism among individual transformations.

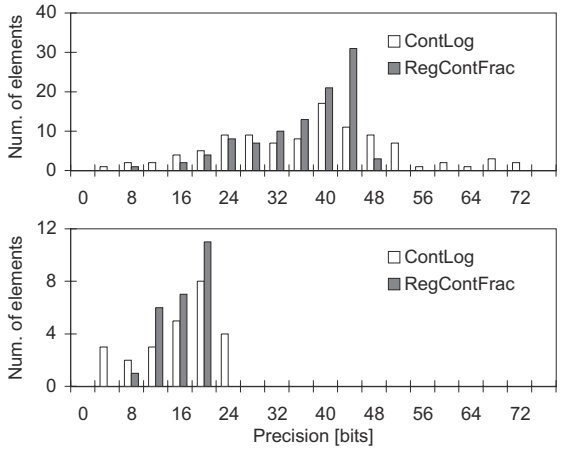
A unique feature of continued logarithms is their use of only powers of two (see (5)), which significantly reduces the complexity of underlying binary arithmetic, i.e. operations performed with fractional coefficients A to H. Where regular continued fractions multiply and divide by general integers, CCL use integer powers of two. That is, continued logarithms reduce all multiplications and divisions to simple shifts while keeping the complexity of addition and subtraction unchanged. In case of the BCL representation, all transformations then use only 1-bit shifts, add/sub and exchange operations (see Tab. 1). Notice that it is namely this property which gives continued logarithms their better potential for direct hardware realization.

2.3. Practical Evaluation

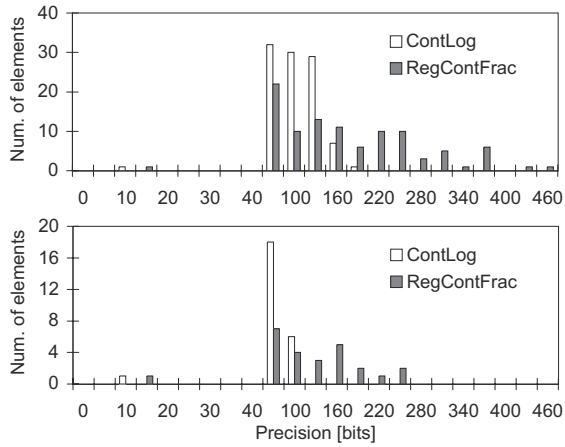
Properties and principles of continued arithmetic makes it promising for implementation of exact arithmetic. Unfortunately, there is a general problem with relatively high complexity of algorithms computing the (bi)linear fractional transformations. This is partially caused by the complexity of arithmetic operations with fractional coefficients, such as multiplication and notably division. As continued logarithms replace these critical operations with relatively simple shifts, performance of their algorithms should increase. On the other hand, continued logarithms have worse approximation convergence than regular continued fractions and it will take them longer to reach a result with a required precision. It is hard to predict, which of these conflicting properties will have greater impact on the overall performance.

To evaluate the effect of continued logarithm properties and to see if they are rather advantageous or not, we compared continued logarithms with regular continued fractions on few practical examples. There were two parameters we observed – performance and precision. Performance is considered here naturally as time to complete the computation and its importance is immediate. The precision parameter represents a peak precision (i.e. bit-length) required for fractional coefficients during the computation. We observed this parameter to get an idea of hardware resource requirements, i.e. register size and data path width, and thus to indicate which of the considered representations would be more effective from hardware design perspective.

For the evaluation purposes, we used Java as an implementation basis and run several simple benchmarks, including random-generated BLFT evaluation, polynomial evaluation and matrix inversion. BLFTs were generated with



(a) Hilbert matrix: 10×10 (top) and 5×5 (bottom)



(b) Random matrix: 10×10 (top) and 5×5 (bottom)

Figure 2. Histogram of fractional coefficients' precision across the computed elements of an inverted matrix.

8-bit signed coefficients and with floating-point arguments in range $[0.0, 1.0]$. Rump's polynomial [12] served as an example of precision sensitive calculation and it was also used to verify accuracy of our implementation. Computing a matrix inverse was finally the most complex benchmark we also used to measure performance. Particular implementations we compared were BCL representation introduced in preceding sections and generic regular continued fractions described in [5].

A complete list of results with detailed comments may be found in [1] and we thus limit here to rephrase only important conclusions. We can generally say that none of both continued representations has an evident advantage over the other one. In the test of random BLFTs, the results were almost equal. Regular continued fractions had average preci-

Table 2. Peak precision of BLFT fractional coefficients detected during computing a result of Rump's polynomial with its precision specified in number of decimal digits.

Result prec. [dec. digits]	1	2	3	4	5	6	7	8	9	10	11	12	13	≥14
Coef. prec. (reg. cont. frac.) [bit]	126	126	126	126	126	126	126	126[1)]	126	126	126	126	126	126
Coef. prec. (BCL) [bit]	170	170	170	170	171	171	171	171	178	178	178	178	345	345[1)]

[1)] At this point an exact solution of a result in a corresponding continued representation has been reached. Note that the solution of Rump's polynomial is a rational number and as such it has a finite representation in both reg. cont. fractions and BCL. To convert this solution into a decimal number system with indicated precision, the continued representation needed to be processed completely (i.e. evaluated to the exact value).

sion per fractional coefficient 51 bits, while continued logarithms required 56 bits. Significant difference showed results of Rump's polynomial evaluation (see Tab. 2). We computed a solution of the polynomial with required precision and observed precision of fractional coefficients during intermediate BLFT calculations. As shown, an approximate solution of the polynomial up to few decimal digits required roughly comparable results (126 and 170 bits per fractional coefficient). However, exact solution in a continued logarithm arithmetic would require 2.5 times more bits per fractional coefficient than regular continued fractions. Finally, the most interesting were results of matrix inversion, partially also because this benchmark was the most complex one. In case of ill-conditioned Hilbert and Pascal matrices, the results came up better for regular continued fractions both in terms of performance and precision of fractional coefficients. On the other hand, inverse of random-generated matrices favored continued logarithms, where performance and especially the precision was remarkably lower. Fig. 2 illustrates these results and displays histograms of fractional coefficient's precision gathered from all elements of an inverse matrix. For instance, we can see that a peak precision for a 10×10 Hilbert matrix was 46 bits for regular continued fractions and 72 bits for continued logarithms. Conversely, a random matrix of the same dimension would require 450 bits and 161 bits, respectively. For comparison of performance refer to [1].

Even though our evaluation had a limited scope, it does not seem so far that there is a major difference between continued logarithms and regular continued fractions. However, continued logarithms theoretically keep their advantage of significantly easier implementation in hardware. With this observation, we had enough justification why to use continued logarithms instead of commonly preferred continued fractions.

3. Arithmetic Unit

In [2] we presented preliminary results of designing an arithmetic unit using continued logarithm representation. In this paper we are about to describe the details of unit's architecture and present the final results of its implementation. We also evaluate performance improvement that such unit could eventually offer.

3.1. Principles

The process of computing BLFT (2) is based on principles of digit-serial on-line computation and interval-like function evaluation. The former principle defines the way of generating output digits serially by having only a partial knowledge about exact values of input arguments. Whenever this partial knowledge is insufficient for producing another output digit, there is possibility to refine it by processing more input digits from one or both arguments. The partial knowledge is specified by intervals restricting the set of possible values that input arguments may gain. With these considerations, it is an immediate consequence that at a given state of the computation we can only determine a range of BLFT functional values and not just a single point. However, this is acceptable as long as the range interval can meet output conditions (1) for continued logarithm digits. Since the exact value of the result is a part of that range, it will certainly meet the output conditions as well.

It follows that the whole process is demand-driven – it takes the actual state (3) of the computation and determines the range of z_i using information on domains of input variables x_k and y_l. Let us denote $\mathbb{Z}_i$ the range of z_i and similarly $\mathbb{X}_k$ and $\mathbb{Y}_l$ the domains of x_k and y_l. If $\mathbb{Z}_i$ cannot satisfy any output condition, input domains are refined by processing more input digits and taking input transformations on fractional coefficients. Thanks to the character of BLFT transformation, $\mathbb{Z}_i$ gets refined too. Once $\mathbb{Z}_i$ is tight enough for output, it is possible to do a z-transformation and produce a corresponding digit. Further digits for $\mathbb{Z}_{i+1}$, $\mathbb{Z}_{i+2}$ and etc. are produced as long as it is possible; after then another input digits are requested and the whole process repeats. This is formally sketched in the algorithm at the end of the paper.

Yet a bit unclear remains determining the range $\mathbb{Z}_i$. In general case, it is a hard and complex task, but in case of well-defined BLFT function, it is possible to take advantage of restrictions on domains $\mathbb{X}_k$ and $\mathbb{Y}_l$. By "well-definedness" we mean that the function is continuous and

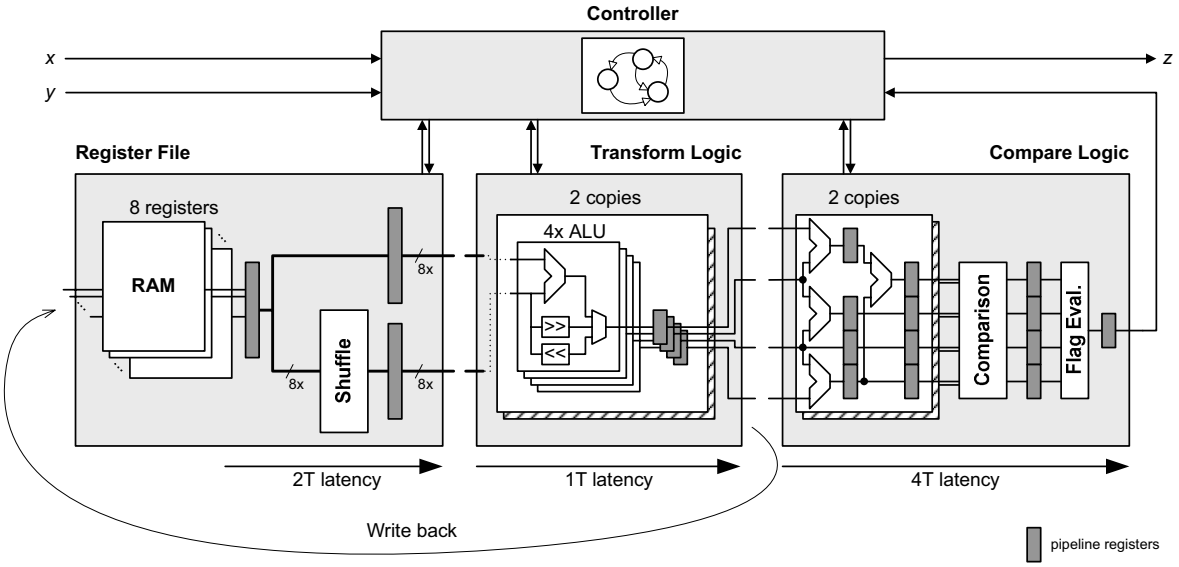


Figure 3. Simplified architecture of the continued logarithm unit.

monotone on its assumed domain. It follows from the definition of BCL (1) that this domain is implicitly restricted to $\mathbb{X}_k \times \mathbb{Y}_l \subseteq [1,\infty]^2$ as it holds $x_k, y_l \geq 1$ for $k, l \geq 2$. Let us define the following *partial sums* among fractional coefficients of BLFT:

$$\begin{aligned} N_0 &= A, & N_2 &= A + C, \\ N_1 &= A + B, & N_3 &= A + B + C + D, \\ D_0 &= E, & D_2 &= E + G, \\ D_1 &= E + F, & D_3 &= E + F + G + H. \end{aligned} \tag{6}$$

One can easily verify that using the implicit restriction to $[1,\infty]^2$ the following conditions are sufficient to make a BLFT well-defined:

$$\begin{aligned} &\operatorname{sgn}(N_0) = \operatorname{sgn}(N_1) = \operatorname{sgn}(N_2) = \operatorname{sgn}(N_3), \\ &\operatorname{sgn}(D_0) = \operatorname{sgn}(D_1) = \operatorname{sgn}(D_2) = \operatorname{sgn}(D_3) \neq 0. \end{aligned} \tag{7}$$

The partial sums actually represent the value of z_i's numerator (N_j) and denominator (D_j) in one of the corners of $[1,\infty]^2$ – e.g. $z_i(\infty, 1) = (A + C)/(E + G) = N_2/D_2$. Therefore, if the "well-definedness" conditions (7) are met, we also check what output conditions (1) the fractions N_j/D_j, $j = 0, \ldots, 3$, satisfy. If they all satisfy the same condition, then z_i's range certainly does too because its BLFT is well-defined. Output conditions may be checked using these equivalences:

$$\begin{aligned} \mathbb{Z}_i \subset [-\infty, 0) &\quad\Leftrightarrow\quad \operatorname{sgn}(N_0) = -\operatorname{sgn}(D_0), \\ \mathbb{Z}_i \subset [0, 1) &\quad\Leftrightarrow\quad 0 \leq N_j < D_j \text{ for } \forall j, \\ \mathbb{Z}_i \subset [1, 2) &\quad\Leftrightarrow\quad D_j \leq N_j < 2D_j \text{ for } \forall j, \\ \mathbb{Z}_i \subset [2, +\infty] &\quad\Leftrightarrow\quad D_j \leq N_j \text{ for } \forall j. \end{aligned} \tag{8}$$

Let us finally note that the introduced principle is deterministic, but it is finite only if input arguments are rational numbers. It is because rational numbers have finite continued logarithm representation and can be thus processed in a finite time, and because their BLFT will produce a rational number too. In general case of real-valued arguments, problems of non-computability may arise because of computing with infinite continued logarithms of irrational numbers. However, we leave this well-known limitation [9] for discussion in Section 4.

3.2. Architecture

The unit's operation follows the process of computing BLFT discussed previously and its architecture thus naturally reflects this character (see Fig. 3). The parts depicted in the figure correspond to individual steps of the computing process, except for added controller that manages the overall operation.

The *controller* takes digits from input arguments and emits computed digits of the result. The core functionality, i.e. the data intensive processing of fractional coefficients, is left to the rest of the unit representing the data path. The controller coordinates the data processing and decides on proceeding of the computation e.g. by following the algorithm discussed in the previous section. For this purpose it relies on status signals of data path modules, and especially those of comparison logic, which provides information on possibility of output.

Values of fractional coefficients are stored in the *register file*, which is composed of eight dual-port RAMs, each one

for storing a single fractional coefficient. The capacity of a RAM module thus defines the maximum precision of a fractional coefficient. The register file also provides the functionality to prepare its output data for further processing, so that computing of cubic transformations can be handled more easily. This is the purpose of the *shuffle block* that can permute register file outputs with respect to a "dimension" (x, y, z) of a selected transformation. If we call the output of the shuffle block *permuted*, then e.g. for y-transformation there will be an eight-tuple (A, B, C, D, E, F, G, H) on the direct output of the register file and the corresponding permuted output will be (B, A, D, C, F, E, H, G).

Both direct and permuted outputs of the register file are further processed by the *transform logic* module, which computes the individual cubic transformations according to a selected dimension. The module is composed of eight ALUs so that i-th ALU takes i-th direct and i-th permuted output. ALUs can be grouped by four, one group for numerator and one for denominator of a computed BLFT. The ALU itself is quite simple and its functionality covers all operations occurring within the cubic transformations (see Tab. 1). These operations include 1-bit shift left/right, addition, subtraction and negation. Important is also the ability of "no operation" used only to propagate permuted outputs so as to realize coefficients exchange such as in case of y-transformation for '/' digit.

New values of fractional coefficients produced by the transform logic are stored back to the register file. At the same time, these values are passed to the module of *compare logic*, which computes the partial sums N_j, D_j defined in (6) and evaluates well-definedness (7) and the output conditions (8). Final decision on taking the z-transformation and producing an output digit is left to the controller.

Although the arithmetic unit operates equally to its sequential software equivalent, the main difference is in employing algorithm-level and "operation"-level parallelism. The algorithm-level parallelism corresponds to parallel computing of the cubic transformations with independent fractional coefficients (Fig. 1). The "operation"-level parallelism means that thanks to the pipelining it is possible to overlap individual steps of computation, such as computing new fractional coefficients together with their partial sums and output evaluation.

Advantage of pipelining and operation overlapping is further supported by unit's multi-cycle character, which means that fractional coefficients are processed by parts (e.g. 32-bit words) rather than at once with full precision. Despite taking more clock cycles to complete a single transformation, the multi-cycle character generally offers better resource utilization and by proper choice of a word size one can leverage different speed/area trade-offs. E.g. a 32-bit unit must iterate sixteen times to compute a transformation of 512-bit fractional coefficients, while a 64-bit unit needs only eight cycles – but the 32-bit unit will most likely be smaller (see Section 3.3 for actual figures). Nonetheless, the primary motivation for introducing the multi-cycle character was to handle extreme precision of thousands bits, which can possibly occur for some ill-conditioned problems, but is rather occasional.

Table 3. Experimental results for different unit's configurations.

Word Size N	**Word Addr.** k	**Complete Precision**	**Freq.**[1)]	**Area**[1)]	**Area**[1,2)]
[bit]	[bit]	[bit]	[MHz]	[Slices]	[%]
8	4	128	218	538	3
16	4	256	205	893	6
32	4	512	180	1 676	12
64	4	1 024	130	3 303	24
32	5	1 024	175	2 113	15
16	6	1 024	200	1 313	9
8	7	1 024	205	1 063	7

[1)] Implementation results obtained from Xilinx ISE 8.1.3 tool.

[2)] The area percentage corresponds to utilization of Xilinx XC2VP30 FPGA device with 13 696 slices.

3.3. Implementation Details

The architecture was described in VHDL hardware description language, with a possibility to configure the word size N (i.e. data path width) and the number of cycles per cubic transformation $M = 2^k$. These parameters account for a total unit's precision of $N \times 2^k$ bits (per fractional coefficient) and their choice also affects area utilization and timing parameters. Such configurability is thus important for evaluating the area/performance trade-offs of a physically realized design.

For implementation we used FPGA (Field Programmable Gate Array) technology because it goes hand-in-hand with the required design-time configurability and gives possibility for future run-time reconfiguration. Important advantage of this technology is immediate availability of silicon devices and thus its potential for rapid prototyping. For purposes of our evaluation, we used Xilinx FPGA devices and in particular Virtex-II Pro family.

Implementation results for few typical configurations of the arithmetic unit are listed in Tab. 3. These results exhibit linear increase of occupied area with respect to data-path width N or address width k, which is a sign of good scalability. Simultaneously with the increasing word size, the frequency partially dropped down as a result of lengthening the critical path going through adders and relating logic. Nonetheless, all configurations were still able to operate at high frequencies. Notice that despite the drop down, using

higher word size still pays off if the associated area overhead is acceptable. Just for convenience we also provide implementation results (Tab. 4) of some highly optimized floating-point units so that one has a fair comparison between the size of a floating-point and the continued logarithm unit.

3.4. Performance Evaluation

Using the presented implementation results, one can estimate[1] the acceleration potential that the proposed arithmetic unit offers in comparison with its software equivalent. Although we base our assessment on very simplified and raw estimates of ideal SW and HW performance, its results will give us some idea of what we may expect when integrating the arithmetic unit as an accelerator in a computer system. For this evaluation we compare a 32-bit variant of the unit with a hypothetical 32-bit RISC-like processor. This means that the fractional coefficients are processed by 32-bit words in both cases.

In case of software implementation, we omit any control overhead and restrict the whole data processing only to major operations within the cubic transformations and $\mathbb{Z}_i$ range evaluation. After analysis of these major operations, we found out that any cubic transformation requires at least four operations and $\mathbb{Z}_i$ range evaluation at least sixteen operations [1]. If we assume each operation to take a single cycle, which is quite realistic, then a minimum time for complete computation:

$$\begin{aligned} T_{\text{min SW}} &= M(K_x + K_y)T_{x/y\text{-transf}} \\ &\quad + MK_z(T_{z\text{-transf}} + T_{\mathbb{Z}\text{ eval}}) \\ &\geq \big(4M(K_x + K_y) + MK_z(4 + 16)\big)T \\ &= \big(4M(K_x + K_y) + 20MK_z\big)T, \end{aligned}$$

where M is the number of words per fractional coefficient and K_x, K_y, K_z are the numbers of digits per continued logarithm representation of individual variables. T is a clock period.

Performance of the arithmetic unit is easy to estimate from its architecture on Fig.3. We see that the latency of individual parts is $L_{\text{Reg.File}} = 2T$, $L_{\text{Transf.Log.}} = T$ and $L_{\text{Cmp.Log.}} = 4T$. If we assume input symbols from both arguments to be instantly available, the arithmetic unit can operate at full bandwidth alternating x- and y-transformations, interleaved by a z-transformation whenever possible. Since the evaluation of output possibility is overlapped with cubic transformations, its overhead projects into only a small constant additive to the complete latency of the arithmetic unit. Performance of the unit is thus:

$$\begin{aligned} T_{\text{HW}} &= (K_x + K_y + K_z)(M + L_{\text{Reg.File}} + L_{\text{Transf.Log.}} \\ &\quad + L_{\text{Cmp.Log.}}) \\ &= (K_x + K_y + K_z)(M + 2 + 1 + 4)T \\ &= (K_x + K_y + K_z)(M + 7)T \end{aligned}$$

Advantage of the hardware implementation is immediate from $T_{\text{min SW}}$ versus T_{HW}. Let us put for simplicity $K_x = K_y = K_z = K$, and we get $T_{\text{min SW}} = 28MK$ for software and $T_{\text{HW}} = 3MK + 21K$ for hardware. Letting $M = 16$ for the total precision of 512 bits, we end up with

$$T_{\text{min SW}} = 448K \quad \text{and} \quad T_{\text{HW}} = 69K,$$

i.e. the arithmetic unit can at least 6.5 times outperform a processor at the same frequency. According to Tab. 3, we may equivalently asset the unit to perform as well as a processor running at 6.5×180 MHz, i.e. above GHz border.

Table 4. Implementation results for some commercially available floating-point units.

Vendor	Architecture	Area [1)] [Slices]	Freq. [MHz]
Gaisler Research [3]	Virtex-II	8 000 (DP)	65
Xilinx [13]	Virtex 4	1 200 (SP)	137.5 ÷ 170

[1)] DP/SP stands for double/single precision.

4. Conclusions

We showed that continued logarithms represent a feasible alternative to commonly preferred regular continued fractions and that principles of their arithmetic can be borrowed for efficient hardware realization. The analysis of the presented results demonstrated that the specifically designed arithmetic unit offers considerably higher performance, i.e. 6.5 times higher, than optimal sequential software implementation. Further improvement is possible by running more such units in parallel. The multi-unit architecture is certainly possible because of fine area utilization, which is in average comparable to that of conventional floating-point units.

There, however, remains the problem of real number computability, which we identified in Section 3.1 and which makes the proposed unit useful only for exact rational arithmetic. Consider a case of an irrational square root with an integer square – continued logarithms fail to compute this square as they cannot produce a finite representation of the integer from an infinite representation of an irrational number. This limitation is a result of the on-line arithmetic character combined with the representation uniqueness [9].

[1]We use this estimation as we did not have real-time performance results at the time of writing.

Nonetheless, there is a chance to find a redundant extension and solve this problem. One possibility is a detection of non-computable cases and their speculative resolution, which can take advantage of the rational arithmetic unit proposed here. The unit would just require some minor extensions of the data path and a change of control mechanism to take care of the speculation. With a relatively low additional overhead, we could finally introduce a continued logarithm unit capable of exact real arithmetic and having implementation parameters similar to those presented here.

Acknowledgment

This work was supported by Czech Technical University under the grant no. CTU0609213. The author would also like to thank anonymous referees for their helpful comments.

References

[1] T. Brabec. On Exact Real Hardware with Specialization to Continued Methods. PhD Proposal, 2006. http://service.felk.cvut.cz/anc/brabect1/pub/phdprop06.pdf.

[2] T. Brabec and R. Lórencz. Arithmetic Unit Based on Continued Fractions. In *Proceedings of the 7th International Scientific Conference on Electronic Computers and Informatics ECI 2006*. http://service.felk.cvut.cz/anc/brabect1/pub/eci06.pdf.

[3] E. Catovic. GRFPU - High Performance IEEE-754 Floating-Point Unit, 2004. http://www.gaisler.com/doc/grfpu_wp.pdf.

[4] P. Flajolet, B. Vallée, and I. Vardi. Continued Fractions from Euclid to The Present Day, 2000.

[5] R. W. Gosper. Continued Fraction Arithmetic. Unpublished manuscript, 1977.

[6] R. W. Gosper, M. Beeler, and R. Schroeppel. HAKMEM. Technical report, Cambridge, MA, USA, 1972. Item 101.

[7] P. Gowland and D. Lester. A Survey of Exact Arithmetic Implementations. In J. Blanck, V. Brattka, and P. Hertling, editors, *CCA*, volume 2064 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2000.

[8] A. Y. Khinchin. *Continued Fractions, 3rd ed.* The University of Chicago Press, 1964.

[9] P. Kornerup and D. W. Matula. An Algorithm for Redundant Binary Bit-Pipelined Rational Arithmetic. *IEEE Trans. Computers*, 39(8):1106–1115, 1990.

[10] M. Niqui. Exact Arithmetic on the Stern-Brocot Tree. Technical Report NIII-R0325, Nijmeegs Instituut voor Informatica en Informateikunde, 2003.

[11] P. J. Potts. *Exact Real Arithmetic using Möbius Transformations*. PhD Thesis, Imperial College, London, 1998.

[12] S. M. Rump. Algorithms for Verified Inclusions – Theory and Practice. *Reliability in Computing: The Role of Interval Methods in Scientific Computing*, pages 109–126, 1988.

[13] Xilinx, Inc. APU Floating-Point Unit v2.1, 2006.

Algorithm: Computing BCL of BLFT

input:	`coeffs` ... coefficients of BLFT, see (2)
	`x` ... *null*-terminated BCL of x variable
	`y` ... *null*-terminated BCL of y variable
output:	`z` ... *null*-terminated BCL of z variable
	`i` ... number of produced BCL digits

```
i = k = l = 0;
x_k = x;    y_l = y;    z_i = null;
Z_i = ∅;    X_k = Y_l = [−∞, +∞];

while (Z_i ≠ {+∞}) {
   Z_i = rangeBLFT(coeffs, X_k, Y_l);
         /* see Eq. (8) and relating */

   switch (Z_i) {
      case Z_i ⊆ [2, +∞] :  digit = '1'; break;
      case Z_i ⊆ [1, 2) :   digit = '0'; break;
      case Z_i ⊆ [0, 1) :   digit = '/'; break;
      case Z_i ⊆ [−∞, 0) :  digit = ' − '; break;
      default :  digit = null;
   }

   if (digit ≠ null) {
      coeffs = transform(coeffs, DIRECTION_z, digit);
         /* see Tab. 1 */
      z_i = digit ∘ z_{i+1};    /* concat.  */
      i++;
      z_i = null;
      continue;
   }

   /* Function select() selects an input arg.
    * for input transformation.  Selection may
    * be random or deterministic - e.g.
    * strict alternation */
   if (select(x_k, y_l) == x_k) {
      digit = getMSD(x_k);
      if (digit == null) continue;
      x_{k+1} = discardMSD(x_k);
         /* rest of BCL repres.  except
          * most significant digit (MSD) */
      coeffs = transform(coeffs, DIRECTION_x, digit);
         /* see Tab. 1 */
      k++;
      X_k = rangeBCL(x_k);
         /* uses implicit restriction to
          * subset of [1,∞]      */
   } else {
      digit = getMSD(y_l);
      if (digit == null) continue;
      y_{l+1} = discardMSD(y_l);
         /* rest of BCL repres.  except MSD */
      coeffs = transform(coeffs, DIRECTION_y, digit);
         /* see Tab. 1 */
      l++;
      Y_l = rangeBCL(y_l);
         /* implicit restriction (see X_k) */
   }
}

z = z_0;
```

Rigorous lower bounds for the topological entropy via a verified optimization technique

Balázs Bánhelyi
University of Szeged
Institute of Informatics
H-6701 Szeged, Hungary
banhelyi@inf.u-szeged.hu

Tibor Csendes
University of Szeged
Institute of Informatics
H-6701 Szeged, Hungary
csendes@inf.u-szeged.hu

Barnabás M. Garay
Budapest University of Technology
Institute of Mathematics
H-1521 Budapest, Hungary
garay@math.bme.hu

Abstract

Our automatic method developed for the detection of chaos is used for finding rigorous lower bounds for the topological entropy of the classical Hénon mapping. We do this within the abstract framework created by Galias and Zgliczynski in 2001, and focus on covering graphs involving different iterations. Our results are compared to those obtained by them.

1. Introduction

Starting from the landmark paper by Mischaikow and Mrozek [11] on the Lorenz equation, computer–assisted proofs for chaos have become an integral part of dynamical systems theory. They upgraded the importance of verified numerics to general mathematics considerably.

What the computer is actually used for is the rigorous checking of a finite number of inclusions of the form

$$\mathcal{T}_j(W_j) \subset U_j \ , \quad j = 1, 2, \ldots, M, \tag{1}$$

where the W_j's and U_j's are subsets of the phase space and the $\mathcal{T}_j$'s are functions associated with the dynamics. The collection of inclusions (1) is structured by the concept of covering relations [16] and forms a sufficient condition for chaos, more precisely, for the embeddability of a certain type of symbolic dynamics into the mapping or differential equation under investigation.

The topological entropy of the embedded symbolic dynamics can be easily determined on the basis of inclusions (1) and is a lower bound for the topological entropy of the mapping or of a Poincaré mapping of the differential equation considered. Computer–assisted proofs for chaos lead to positive lower bounds for topological entropy. Positive topological entropy is an important qualitative indicator for chaos. The larger the topological entropy the stronger chaos. For diffeomorphisms, positive topological entropy implies positivity of the maximal Lyapunov exponent.

The real task is not to check inclusions (1) but to find the W_j's, U_j's, and $\mathcal{T}_j$'s involved in a sufficient condition for chaos.

In a recent paper of ours [4], this task is formulated as a constraint satisfaction problem in optimization theory and thus the possibility of applying algorithms of global optimization in computer–assisted proofs for chaos opened. As for the first application, we reproved Zgliczynski's result [16] on embeddability of the full shift on two symbols—shortly: the existence of Σ_2–chaos—in the seventh iterate of the widely investigated Hénon mapping

$$\mathcal{H} : \mathbb{R}^2 \to \mathbb{R}^2 \ , \ (x, y) \to (1 + y - ax^2, bx)$$

with the classical parameters

$$a = 1.4 \ \text{ and } \ b = 0.3 \, .$$

Our proof in [4] differs strikingly from all computer–assisted proofs for chaos (we mean those based on embedding symbolic dynamics we are aware of): There is no trial and error interaction between computer and computer scientist, and the tedious task of adjusting the distinguished sets W_j, U_j in (1) by hand is left entirely to the computer. Nevertheless, a reasonable choice of the search domain in the optimization procedure requires a "good initial guess" for the W_j's and U_j's exploiting a priori numerical and theoretical results on the dynamics. This is the only reason why our method cannot be termed fully automatic.

The present paper is a continuation of our previous work [4]. We focus on computing lower bounds for the topological entropy of the classical Hénon mapping based on covering graphs involving different iterations.

Though they are valid in a much greater generality [10], [12], all abstract results in Section 2 are formulated only in two dimension. We follow the presentation in Galias and

Zgliczynski [9] very closely. Covering relation and topological entropy, the two most relevant notions of dynamical systems theory are defined and thoroughly discussed in Subsections 2.1 and 2.2, respectively. We assume that the reader is (at least, intuitively) familiar with Smale's classical horseshoe presented in Figure 4(a), a well–known example for chaos. The type of combinatorial and dynamical complexity of Smale's classical horseshoe is termed as Σ_2–chaos. For dynamical systems theory in general, we refer to [14].

Section 3 is devoted to describing the interval arithmetic based checking algorithm and the optimization model for the collection of inclusions (1).

In Section 4 applications to the classical Hénon mapping are presented. The compact set in (2) is chosen for the standard trapping region, a positively invariant trapezoid for $\mathcal{H}$. It is actually the continuous mapping $\varphi = \mathcal{H}|X$, the restriction of $\mathcal{H}$ to X to which the abstract results of Section 2 and the computational methods of Section 3 are applied. Our results are compared to those obtained by Galias and Zgliczynski [6, 8, 9, 16, 17]. The greater part of them confirms earlier intuition but a small part of them gives new geometric insight on the existence of Σ_2–chaos for small and large iterates of $\mathcal{H}$.

We cannot conceal that the combination of what are called "educated guesses" and computer experimentation with human overhead is still better. The 2002 lower bound of Galias [8] for the topological entropy of the classical Hénon mapping remains the best known to date. Nevertheless, we think that automatic methods for rigorous chaos detection – including those that lead to rigorous estimates for quantitative chaos indicators – have a huge potential for the future and, eventually, from the view–point of effectiveness, they supersede those requiring repeated human interference. In particular, with further improvements, we hope that the model case applications of our verified optimization method [4] in Section 4 to bound topological entropy (the only new results of the present paper) will be competitive to those governed by hands.

2. Some abstract results in advance

2.1 The covering relation

Throughout this paper, let $Q_1, Q_2, \ldots, Q_N$ denote pairwise disjoint, closed, solid quadrangles in $\mathbb{R}^2$ with pairs of opposite edges termed horizontal and vertical. Continuing the vertical edges by parallel half–lines, two closed, solid, vertical stripes—an upper stripe and a lower stripe—are attached to each quadrangle. The upper stripe and the quadrangle share the upper horizontal edge whereas the lower stripe and the quadrangle share the lower horizontal edge of the quadrangle. Together with the attached stripes, each quadrangle separates $\mathbb{R}^2$ to a left and a right part. They are open, topological half–planes.

For $i = 1, 2, \ldots, N$, the union of the two attached stripes is denoted by E_i. Left and right vertical edges of Q_i are denoted by e_i^L and e_i^R, respectively. The corresponding left and right topological half–planes are components of the set $\mathbb{R}^2 \setminus (Q_i \cup E_i)$. They are denoted by $\mathcal{O}_i^L$ and $\mathcal{O}_i^R$, respectively.

Let X be a compact set with the property

$$\bigcup_{i=1}^{N} Q_i \subset X \subset \mathbb{R}^2, \tag{2}$$

and consider a continuous mapping $\varphi : X \to X$.

Definition 1 *Following Galias and Zgliczynski* [9]*, we say that Q_i φ^k–covers Q_j and use the notation $Q_i \overset{k}{\Longrightarrow} Q_j$ ($k = k(i,j)$ is a positive integer) if*

(i) *the image of Q_i under φ^k is located between the horizontal edges of Q_j or, equivalently,*

$$\varphi^k(Q_i) \subset \mathbb{R}^2 \setminus E_j$$

and

(ii) *the images of the vertical edges of Q_i under φ^k have empty intersection with $Q_j \cup E_j$ and they are located on the opposite sides thereof. In other words, one of the following two alternatives holds true: Either*

$$\varphi^k(e_i^L) \subset \mathcal{O}_j^L \text{ and } \varphi^k(e_i^R) \subset \mathcal{O}_j^R$$

(see Figure 1) or

$$\varphi^k(e_i^L) \subset \mathcal{O}_j^R \text{ and } \varphi^k(e_i^R) \subset \mathcal{O}_j^L\ .$$

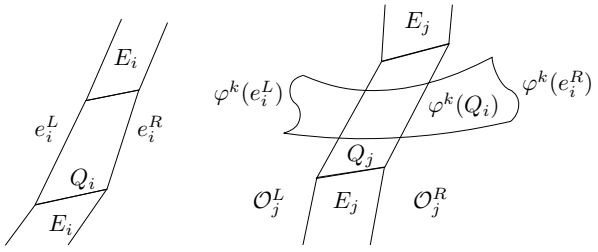


Figure 1. The covering properties

Definition 1 leads naturally to the concept of the covering graph. The covering graph is a directed graph with vertices $Q_1, Q_2, \ldots, Q_N$ and weighted edges. The pair (Q_i, Q_j) belongs to the edge set of the covering graph if $Q_i \overset{k}{\Longrightarrow} Q_j$ for some positive integer k and integer k is the weight on

this edge. Thus the covering relation $Q_i \overset{k}{\Longrightarrow} Q_j$ represents a weighted edge of the covering graph and vice versa. Multiple edges have different weights. Loop edges are also allowed. The *covering matrix* is an $N \times N$ matrix C with elements $\{c_{ij}\}_{i,j=1}^{N}$ where $c_{ij} = k > 0$ if $Q_i \overset{k}{\Longrightarrow} Q_j$ for a unique $k = k(i,j)$ and 0 otherwise. Thus the covering matrix is defined only in the absence of multiple edges.

Of course everything depends on the choice of the pairwise disjoint quadrangles $Q_1, Q_2, \ldots, Q_N$. The more careful this choice, the richer the structure of the covering graph can be, and the higher the estimate for $h(\varphi)$, the topological entropy of mapping φ.

The *expanded covering graph* is defined as follows. For $i = 1, 2, \ldots, N$, set

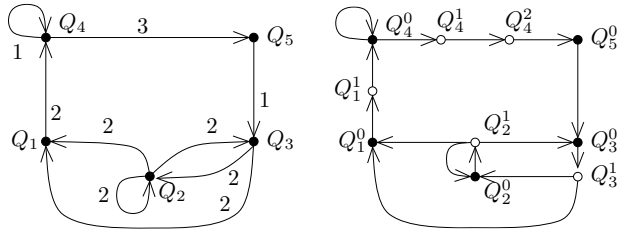
$$s(i) = \max\{k \mid Q_i \overset{k}{\Longrightarrow} Q_j \ , \ j = 1, 2, \ldots, N\} \, ,$$

the maximum weight on outgoing edges at Q_i. In case there are no outgoing edges from Q_i, we take $s(i) = 0$. If $0 \leq s(i) < 2$, vertex Q_i is renamed as Q_i^0. If $s(i) \geq 2$, then vertex Q_i is replaced by the string of $s(i)$ vertices and the connecting $s(i) - 1$ directed edges

$$Q_i^0 \to Q_i^1 \to \ \ldots \ \to Q_i^{s(i)-1} \, .$$

In both cases, vertex Q_i^0 of the expanded covering graph is identified with vertex Q_i of the covering graph. For $i \in \{1, 2, \ldots, N\}$ with $s(i) \geq 2$, the new, intermediate vertices $Q_i^1, Q_i^2, \ldots, Q_i^{s(i)-1}$ of the expanded covering graph are identified with the sets $\varphi(Q_i), \varphi^2(Q_i), \ldots, \varphi^{s(i)-1}(Q_i)$, respectively. Finally, edge $Q_i \overset{k}{\Longrightarrow} Q_j$ of the covering graph is replaced by the 'not–in–the–string' edge $Q_i^{k-1} \to Q_j^0$ of the expanded covering graph. For an example, see Figure 2.

The *expanded covering matrix* A is defined as the adjacency matrix of the expanded covering graph. In other words, A is the $N^E \times N^E$ 0–1 matrix with $a_{pq} = 1$ if there is an edge of the expanded covering graph starting from the p–th and arriving at the q–th vertex and 0 otherwise. Note that N^E is the sum of N (the number of the original vertices of the covering graph) and of $\sum_i \{(s(i) - 1) \mid s(i) \geq 2\}$ } (the number of the new, intermediate vertices).



(a) A covering graph (b) Its expanded version

Figure 2. Expanding covering graphs

The most important property of the concept of covering is summarized in the following fundamental result stating that paths and circles of the covering graph can be shadowed by true trajectories.

Theorem 1 [9] *Suppose we are given a finite chain of coverings*

$$Q_{i_0} \overset{k_0}{\Longrightarrow} Q_{i_1} \overset{k_1}{\Longrightarrow} \ \ldots \ \overset{k_M}{\Longrightarrow} Q_{i_{M+1}}. \tag{3}$$

For brevity, we write $\ell_0 = 0$ and, inductively, $\ell_{m+1} = \ell_m + k_m$, $m = 0, 1, \ldots, M$. Then there exists a finite sequence of points

$$x_{\ell_0} \in Q_{i_0}, x_{\ell_1} \in Q_{i_1}, \ldots, x_{\ell_{M+1}} \in Q_{i_{M+1}}$$

such that

$$\varphi^{\ell_m}(x_0) = x_{\ell_m}, \ m = 0, 1, \ldots, M+1.$$

If, in addition,

$$i_0 = i_{M+1},$$

then

$$\big(Q_{i_0} = Q_{i_{M+1}} \ and \ \big) \ x_{\ell_0} = x_{\ell_{M+1}}.$$

For $s = 0, 1, \ldots, k_m - 1$ and $m = 0, 1, \ldots, M$, define $x_{\ell_m + s} = \varphi^{\ell_m + s}(x_0)$ and observe that $x_{\ell_m+s} \in Q_{i_m}^s = \varphi^s(Q_{i_m})$. Note also that $x_{j+1} = \varphi(x_j)$ whenever $j = 0, 1, \ldots, \ell_{M+1} - 1$. Thus we are justified in saying that $\{x_j\}_{j=0}^{\ell_{M+1}}$ is a *φ–trajectory segment induced by the chain of coverings* (3). The length of this trajectory segment is $L = \ell_{M+1}$. The distance between two trajectory segments $\{x'_j\}_0^L$ and $\{x''_j\}_0^L$ of equal length is $\max_j |x'_j - x''_j|$.

2.2 Symbolic dynamics and topological entropy

For convenience, we recall here the definition of the topological entropy. As before, we assume that X is a compact set with property (2) and function $\varphi : X \to X$ is continuous.

Definition 2 *For $n = 1, 2, \ldots$ and $\varepsilon > 0$, a (necessarily finite) set $Y \subset X$ is called (n, ε)–separated if for every two different points $y, \tilde{y} \in Y$, there exists an integer $j \in [0, n)$ such that*

$$|\varphi^j(y) - \varphi^j(\tilde{y})| > \varepsilon \, .$$

With $\sharp Y$ denoting the cardinality of Y, set

$$s_n(\varepsilon) = \max \{ \, \sharp Y \mid Y \subset X \ \textit{is} \ (n, \varepsilon)\textit{-separated} \, \} \, .$$

Finally, the topological entropy of φ is defined as

$$h(\varphi) = \lim_{\varepsilon \to 0} \limsup_{n \to \infty} \frac{1}{n} \log s_n(\varepsilon) \, .$$

Roughly speaking, topological entropy measures the scaled number of metrically different trajectory segments of increasing length. ($s_n(\varepsilon)$ itself means the maximum number of discrete points that can be packed into X before there exists two of them that do not separate by ε after n iterates.)

Theorem 2 [9] *Using the terminology adopted in the last paragraph of the previous subsection, assume that*

(a) *for some $\varepsilon_0 > 0$, the distance between any two φ–trajectory segments of equal length which are induced by different chains of coverings is at least ε_0.*

Then

$$h(\varphi) \geq \log \lambda_1(A),$$

where $h(\varphi)$ is the topological entropy of φ and $\lambda_1(A)$ is the dominant eigenvalue of the expanded covering matrix.

Note that assumption (a) is considerably weaker than each of the standard assumptions

(b) all iterates involved in the covering graph are the same, i.e., all edges of the covering graph have the same weight, say k^*

or

(c) the original quadrangles $\{Q_i^0\}_{i=1}^N$ and the new, intermediate sets $\{Q_i^1, Q_i^2, \ldots, Q_i^{s(i)-1}\}_{\{i \mid s(i)\geq 2\}}$ altogether are pairwise disjoint.

The dominant eigenvalue $\lambda_1(A)$ satisfies inequality $\lambda_1(A) \geq |\lambda_j(A)|$ for all eigenvalues of A. The existence of such an eigenvalue is part of the Frobenius–Perron theorem on nonnegative matrices [14]. Note that $\lambda_1(A) > 1$ if and only if the covering graph has two different but intersecting directed circles (a loop edge is not excluded). In view of Theorem 2, inequality $\lambda_1(A) > 1$ is a sufficient condition for chaos.

It is worth mentioning here that assumption (b) implies

$$\lambda_1(C) = k^* \left(\lambda_1(A)\right)^{k^*}$$

for the dominant eigenvalue of the covering matrix. On the other hand, assumption (c) implies that φ is semiconjugate to a subshift of finite type with transition matrix A. None of these two latter properties is a consequence of assumption (a).

The crucial task is to guarantee that assumption (a) is satisfied. The problem is to control the position of the intermediate sets. It is clearly enough to check the existence of a positive integer T with the property as follows.

Property (P) *Given any two φ–trajectory segments $\{x'_j\}_0^L$ and $\{x''_j\}_0^L$ of equal length*

$$L = k'_0 + k'_1 + \ldots + k'_{M'} = k''_0 + k''_1 + \ldots + k''_{M''}$$

and which are induced by the different chains of coverings

$$Q_{i'_0} \overset{k'_0}{\Longrightarrow} Q_{i'_1} \overset{k'_1}{\Longrightarrow} \ldots \overset{k'_{M'}}{\Longrightarrow} Q_{i'_{M'+1}}$$

and

$$Q_{i''_0} \overset{k''_0}{\Longrightarrow} Q_{i''_1} \overset{k''_1}{\Longrightarrow} \ldots \overset{k''_{M''}}{\Longrightarrow} Q_{i''_{M''+1}},$$

there exist integers $n \in [0, L]$ and $t \in [0, T]$ such that

$$x'_n \in Q_{i'_p} \quad \textit{for some integer} \quad p \in [0, M'+1]\,, \tag{4}$$

$$x''_n \in \varphi^t(Q_{i''_r}) \quad \textit{for some integer} \quad r \in [0, M''+1] \tag{5}$$

with $n = \#\left\{\textit{iterates up to } Q_{i''_r}\right\} + t$ and

$$Q_{i'_p} \cap \varphi^t(Q_{i''_r}) = \emptyset, \tag{6}$$

or, alternatively,

$$x'_n \in \varphi^t(Q_{i'_p}) \quad \textit{for some integer} \quad p \in [0, M'+1]\,, \tag{7}$$

$$x''_n \in Q_{i''_r} \quad \textit{for some integer} \quad r \in [0, M''+1] \tag{8}$$

with $n = \#\left\{\textit{iterates up to } Q_{i'_p}\right\} + t$ and

$$\varphi^t(Q_{i'_p}) \cap Q_{i''_r} = \emptyset. \tag{9}$$

If property (P) is satisfied, ε_0 in assumption (a) can be chosen as

$$\varepsilon_0 = \min\{\text{ distance}(Q_i, \varphi^t(Q_j))\ \},$$

where the minimum is taken for all $i, j = 1, 2, \ldots, N$ and $t = 0, 1, \ldots, T$ with $Q_i \cap \varphi^t(Q_j) = \emptyset$.

3. Computer procedures to analyze the covering property

In order to check subset relations of the form $\mathcal{T}(W) \subset U$ in a rigorous way, several algorithms were developed in the last decade. They form an integral part of what is called set–valued numerics and are surveyed in [5]. The key task is, however, to establish the subset relations themselves. We assume that $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$ is continuous, $W \subset \mathbb{R}^n$ is compact, $U \subset \mathbb{R}^n$ is open. The major assumption is that $\mathcal{T}$, W, and U depend on some vector $\lambda \in \Lambda$ of parameters where Λ is a compact subset of $\mathbb{R}^m$. The parameter vector λ_0 has to be specified in such a way that the resulting subset relation $\mathcal{T}(\lambda_0)(W(\lambda_0)) \subset U(\lambda_0)$ is fulfilled.

In what follows the task of finding successful subset relations is modeled as a constrained optimization problem. The checking algorithm presented first will then be built in a framework optimization algorithm and numerical results provided.

3.1 A checking algorithm

The checking algorithm is a branch–and–bound procedure using interval arithmetic based inclusion functions [1, 13]. The point–to–point transformation $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$ is replaced by its natural interval extension $T : \mathbb{I}^n \to \mathbb{I}^n$ where $\mathbb{I}^n$ stands for the set of all closed and axis-aligned rectangles in $\mathbb{R}^n$. Note that $\mathcal{T}(x) \in T(I)$ whenever $I \in \mathbb{I}^n$ with $x \in I$. For $I, J \in \mathbb{I}^n$, $I \subset J$ implies $T(I) \subset T(J)$. The width of the rectangle I for which the i-th component is $[\underline{x}_i, \overline{x}_i]$ is defined as $w(I) = \max\{|\overline{x}_i - \underline{x}_i| \,\big|\, i = 1, 2, \ldots, n\}$. For any bounded subset S in $\mathbb{R}^n$, note that $w(T(I_j)) \to 0$ holds for all interval sequences $\{I_j\}$ with $I_j \subset S$ for all $j = 1, 2, \ldots$ and $w(I_j) \to 0$. To enclose the rounding errors and to provide verified numerical results we use the outward rounding that gives computer representable result intervals containing all the points of the real operations.

ALGORITHM 1 *The Checking Routine*

Inputs:
- ε: the user set limit size of subintervals,
- W: the argument set,
- U: the aimed set for which $\mathcal{T}(W) \subset U$ is to be checked.

1. Calculate the initial interval $I \supset W$
2. Push the initial interval into the stack
3. **while** (the stack is nonempty)
4. Pop an interval I out of the stack
5. Calculate the transformed interval $J = T(I)$
6. **if** $I \cap W \neq \emptyset$, and the condition $J \subset U$ does not hold, **then**
7. **if** the width of interval I is less than ε **then**
8. push the subintervals into the output list ($\mathcal{J}$)
9. **else** bisect I along the widest side: $I = I_1 \cup I_2$
10. push the subintervals into the stack
11. **endif**
12. **endif**
13. **end while**
14. **print** that $\mathcal{T}(W) \subset U$ is proven and **stop**

For details as well as for a formal proof of the correctness of Algorithm 1, see our paper [4].

3.2 The accompanying optimization problem

Each relation $\mathcal{T}_j(W_j) \subset U_j$, $j = 1, 2, \ldots, M$ is analyzed separately. The j–th execution of Algorithm 1 may result in an interval $I_j = I_j(\lambda)$ such that $I_j \cap W_j \neq \emptyset$ but $T_j(I_j) \subset U_j$ does not hold true. This means that the j–th execution of Algorithm 1 ends at Step 8 — let $\mathcal{J} = \mathcal{J}(\lambda) = \{j_1, \ldots, j_\ell\} \subset \{1, 2, \ldots, M\}$ denote the set of such indices. Otherwise, for $j \notin \{j_1, \ldots, j_\ell\}$, the j–th execution of Algorithm 1 ends at Step 14.

Consider the optimization problem

$$\min_{\lambda \in \Lambda} g(\lambda)$$

where

$$g(\lambda) = p\left(\sum_{j \in \mathcal{J}(\lambda)} \max_{v \in T_j(I_j(\lambda))} \inf_{u \in U_j(\lambda)} |u - v|\right), \quad (10)$$

with $p(r) = r + 1$ if r is positive and $p(r) = 0$ otherwise. Here $I_j(\lambda)$ is the interval returned by the checking routine for $j \in \mathcal{J}(\lambda)$ (and the empty set for $j \notin \mathcal{J}(\lambda)$), $\Lambda \subset \mathbb{R}^m$ is the search set (the compact set of admissible parameter values), and $p : \mathbb{R} \to \mathbb{R}$ is the penalty function. Note that the j's summation term $\max_{v \in T_j(I_j(\lambda))} \inf_{u \in U_j(\lambda)} |u - v|$ of the argument of the penalty function in (10) is a nondifferentiable function of λ and stands for the Hausdorff distance of the transformed subinterval $T_j(I_j(\lambda))$ to the set $U_j(\lambda)$, a value proportional to the measure of how much condition $J \subset U$ (i.e. condition $T_j(I_j(\lambda)) \subset U_j(\lambda)$, $j \in \mathcal{J}(\lambda)$ in Step 6 of Algorithm 1) is violated.

The computation of this Hausdorff distance requires some geometry. It is an elementary task provided that $n = 2$ and that the boundary of each $U_j(\lambda)$ consists of a moderate number of finite or infinite straight line segments.

The penalty function p adds a fixed penalty term in case at least one of the constraints is not satisfied. Hence, if an optimization algorithm leads to a parameter vector λ_0 with $g(\lambda_0) = 0$, then – at the same time – the built-in checking routine provides a guaranteed reliability computational proof of the respective subset relations $\mathcal{T}_j(\lambda_0)(W_j(\lambda_0)) \subset U_j(\lambda_0)$, $j = 1, 2, \ldots, M$. Unfortunately, due to the high degree of nonlinearity of the problem, it is well possible that the output of the optimization algorithm is inconclusive, even if $\min_{\lambda \in \Lambda} g(\lambda) = 0$.

The emerging global optimization problem has been solved by a reliable clustering stochastic optimization method which goes back to [3]. This method is able to find all global optimizer points in search domains of moderate dimension and does not use the differentiability of the objective function. For a detailed discussion of this optimization model and of the relevant techniques of global optimization, see our paper [4].

In all the applications below, a typical parameter is a coordinate of a vertex of a quadrangle. The search domains for the coordinates of the individual vertices are suggested by the position of the periodic points of the Hénon map [7] as well as by basic facts on homoclinic saddles.

4. Applications

4.1 A local improvement

Galias and Zgliczynski [9] considered the following configuration of five quadrangles

$$Q_i = \mathrm{conv}\{V_{ul}^{Q_i}, V_{ur}^{Q_i}, V_{ll}^{Q_i}, V_{lr}^{Q_i}\}\ ,\ \ i = 1,2,3,4,5,$$

the closed convex hulls of their respective upper left, upper right, lower left, and lower right vertices. The coordinates of these vertices are

$$\begin{array}{rclrcl}
V_{ul}^{Q_1} &=& (-0.95, 0.39)\,, & V_{ur}^{Q_1} &=& (-0.81, 0.38)\,,\\
V_{ll}^{Q_1} &=& (-0.95, 0.29)\,, & V_{lr}^{Q_1} &=& (-0.77, 0.28)\,,\\
V_{ul}^{Q_2} &=& (-0.80, 0.39)\,, & V_{ur}^{Q_2} &=& (-0.31, 0.34)\,,\\
V_{ll}^{Q_2} &=& (-0.75, 0.29)\,, & V_{lr}^{Q_2} &=& (-0.22, 0.24)\,,\\
V_{ul}^{Q_3} &=& (0.24, 0.30)\,, & V_{ur}^{Q_3} &=& (0.44, 0.25)\,,\\
V_{ll}^{Q_3} &=& (0.07, 0.20)\,, & V_{lr}^{Q_3} &=& (0.36, 0.15)\,,\\
V_{ul}^{Q_4} &=& (0.638, 0.28)\,, & V_{ur}^{Q_4} &=& (0.72, 0.28)\,,\\
V_{ll}^{Q_4} &=& (0.518, 0.07)\,, & V_{lr}^{Q_4} &=& (0.62, 0.07)\,,\\
V_{ul}^{Q_5} &=& (0.74, 0.23)\,, & V_{ur}^{Q_5} &=& (0.88, 0.21)\,,\\
V_{ll}^{Q_5} &=& (0.70, 0.12)\,, & V_{lr}^{Q_5} &=& (0.85, 0.10)\,.
\end{array}$$

All these data were found by human experimentation.

The covering graph and the expanded covering graph are those shown in Figure 2.

All the covering relations and properties

$$\mathcal{H}(Q_4) \cap Q_5 = \emptyset\,, \tag{11}$$

$$Q_5 \cap \mathcal{H}(Q_5) = \emptyset \tag{12}$$

were checked by interval computation. Properties (11) and (12) are crucial in proving that property (P) is fulfilled. In fact, since all the outgoing edges from Q_1, Q_2, and Q_3 have the same weight, it is enough to consider φ–trajectory segments $x'_0, x'_1, \ldots$ and $x''_0, x''_1, \ldots$ induced by the different chains of coverings $Q_4 \overset{1}{\Longrightarrow} Q_4 \Longrightarrow \ldots$ and $Q_4 \overset{3}{\Longrightarrow} Q_5 \Longrightarrow \ldots$, respectively. If

$$Q_4 \overset{1}{\Longrightarrow} Q_4 \overset{1}{\Longrightarrow} Q_4 \Longrightarrow \ldots\ , \tag{13}$$

then $x'_3 \in \mathcal{H}(Q_4)$ by $x'_2 \in Q_4$. But $x''_3 \in Q_5$ and recall (11). If

$$Q_4 \overset{1}{\Longrightarrow} Q_4 \overset{3}{\Longrightarrow} Q_5 \Longrightarrow \ldots\ , \tag{14}$$

then $x'_4 \in Q_5$. But $x''_4 \in \mathcal{H}(Q_5)$ by $x''_3 \in Q_5$ and recall (12). We see that, in both cases, relations (4)–(6) resp. (7)–(9) are satisfied. Thus Theorem 2 applies and yields that [9]

$$h(\mathcal{H}) > 0.338\,.$$

The distinction between subcases (13)–(14) is necessary because the *direct way* is blocked by $\mathcal{H}^2(Q_4) \cap Q_5 \neq \emptyset$.

Keeping Q_1, Q_2, Q_3, and Q_5 fixed, we look for a new quadrangle $\tilde{Q}_4$ so that $\tilde{Q}_4 \overset{2}{\Longrightarrow} Q_5$ and thus the modified expanded covering graph has only a single new vertex between $\tilde{Q}_4$ and Q_5. In other words, we try to replace element $c_{45} = 3$ of the original covering matrix

	Q_1	Q_2	Q_3	Q_4	Q_5
Q_1				2	
Q_2	2	2	2		
Q_3	2	2			
Q_4				1	**3**
Q_5			1		

.

by $\tilde{c}_{45} = 2$. If also property (cf. (11))

$$\mathcal{H}(\tilde{Q}_4) \subset \mathbb{R}^2 \setminus Q_5 \tag{15}$$

is fulfilled, then Theorem 2 applies in a *direct way* and yields that

$$h(\mathcal{H}) > 0.357\,,$$

the logarithm of the dominant eigenvalue of the modified expanded covering matrix $\tilde{A}$.

We are facing an optimization problem with constraints (15) and $Q_1 \overset{2}{\Longrightarrow} \tilde{Q}_4, \tilde{Q}_4 \overset{1}{\Longrightarrow} \tilde{Q}_4, \tilde{Q}_4 \overset{2}{\Longrightarrow} Q_5$. The tangent of the slope of the parallel half–lines determining union $\tilde{E}_4$ of the two attached stripes is taken for 10. This particular choice for the tangent was motivated by our earlier experience [2]. Thus we have eight parameters, the horizontal and the vertical coordinates of the four vertices of quadrangle $\tilde{Q}_4$. Fortunately, the optimization was successful and the vertices of the modified quadrangle $\tilde{Q}_4$ are

$$\begin{array}{rclrcl}
V_{ul}^{\tilde{Q}_4} &=& (0.638, 0.28)\,, & V_{ur}^{\tilde{Q}_4} &=& (0.75, 0.28)\,,\\
V_{ll}^{\tilde{Q}_4} &=& (0.518, 0.07)\,, & V_{lr}^{\tilde{Q}_4} &=& (0.65, 0.07)
\end{array}$$

– we rounded the nine–digit numbers supplied by the computer and checked the constraints again.

4.2 A global search

Now we are looking for the quadrangles Q_1, Q_2, Q_3, Q_4 with the covering matrix

	Q_1	Q_2	Q_3	Q_4
Q_1	1	2		
Q_2			2	2
Q_3			2	2
Q_4	2	2		

.

In order to apply Theorem 2, we add inclusion (cf. (15))

$$\mathcal{H}(Q_1) \subset \mathbb{R}^2 \setminus Q_2$$

as a constraint to those determined by the covering relations.

As before, the tangent of the slope of the parallel half–lines is taken for 10. The vertical coordinates of the 16 vertices of the four quadrangles are also fixed. They are chosen according to the position of the unstable manifold of the famous homoclinic saddle point $P = (0.631\ldots, 0.189\ldots)$ [14] of $\mathcal{H}$. This is an example of how a priori knowledge on the geometry helps to decrease the number of parameters in the relevant constraint satisfaction problem. More explanation will be given in Subsection 4.3 below.

We were lucky again. The solution

$$\begin{array}{lcllcl}
V_{ul}^{Q_1} &=& (0.52, 0.30)\,, & V_{ur}^{Q_1} &=& (0.78, 0.30)\,,\\
V_{ll}^{Q_1} &=& (0.38, 0.05)\,, & V_{lr}^{Q_1} &=& (0.65, 0.05)\,,\\
V_{ul}^{Q_2} &=& (0.24, 0.30)\,, & V_{ur}^{Q_4} &=& (0.51, 0.30)\,,\\
V_{ll}^{Q_2} &=& (0.07, 0.20)\,, & V_{lr}^{Q_4} &=& (0.33, 0.13)\,,\\
V_{ul}^{Q_3} &=& (-0.62, 0.34)\,, & V_{ur}^{Q_2} &=& (-0.38, 0.34)\,,\\
V_{ll}^{Q_3} &=& (-0.51, 0.24)\,, & V_{lr}^{Q_2} &=& (-0.20, 0.24)\,,\\
V_{ul}^{Q_4} &=& (-0.92, 0.38)\,, & V_{ur}^{Q_3} &=& (-0.67, 0.38)\,,\\
V_{ll}^{Q_4} &=& (-0.89, 0.28)\,, & V_{lr}^{Q_3} &=& (-0.63, 0.28)
\end{array}$$

given by the computer is illustrated in Figure 6.

The characteristic polynomial of the expanded covering matrix is

$$p(\lambda) = \lambda^8 - \lambda^7 - \lambda^6 + \lambda^5 - \lambda^4 + \lambda^3 - \lambda^2, \qquad (16)$$

and the accompanying entropy estimate is

$$h(\mathcal{H}) > 0.382. \qquad (17)$$

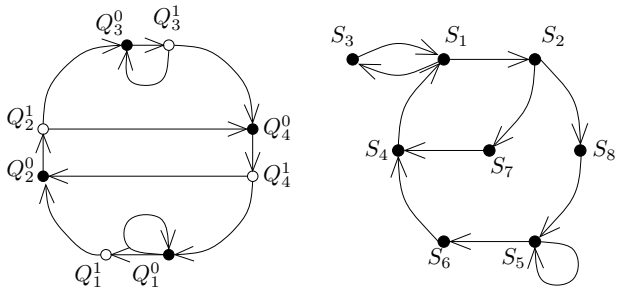
The expanded covering graph of the four–quadrangle configuration found by the computer is shown in Figure 3(a).

The (expanded) covering graph of the eight–quadrangle configuration found in Galias [8] by hand is shown in Figure 3(b). Note that assumption (b) is satisfied in Galias's example with $k^* = 1$. Hence the (expanded) covering matrix is actually the transition matrix of the embedded shift dynamics. Remarkably, the characteristic polynomial is (16) again. This means we could reproduce Galias's estimate (17) by using only four quadrangles, the half of the number of the quadrangles he needed.

4.3 On the underlying geometry

The first global search we completed successfully is illustrated in Figure 5. We looked for two quadrangles with the covering matrix

	Q_1	Q_2
Q_1	2	2
Q_2	2	2

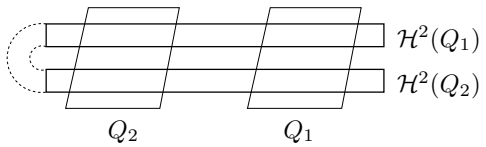


(a) Found by the computer (b) Found by Galias

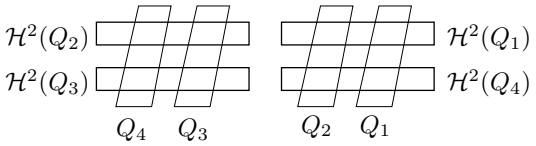
Figure 3. Expanded covering graphs with the same characteristic polynomial

and had 16 parameters again, this time the vertical and the horizontal coordinates of the eight vertices. (As before, the tangent of the slope of the parallel lines was taken to be 10.) The positive outcome of the optimization procedure means embeddability of the full shift on two symbols—shortly: the existence of Σ_2–chaos—in the second iterate of $\mathcal{H}$.

Analogies to the first steps of constructing the maximal invariant set in Smale's classical, piecewise affine horseshoe [14] and the strong resemblance of Figure 5 to the schematic picture of the piecewise linear horseshoe in Figure 4(a) suggests we might look for the four–quadrangle configuration shown schematically in Figure 4(b).



(a) Full shift on two symbols



(b) Towards full shift on four symbols

Figure 4. Piecewise affine horseshoes

But this would only repeat the entropy estimate

$$h(\mathcal{H}) \geq 2^{-1} \log 2 > 0.346$$

obtained on the basis of Figure 5. The possibility of improving the $2^{-1}\log 2$ lower bound is opened via replacing the $c_{11} = 2$ element of the covering matrix that corresponds to Figure 4(b) by $\tilde{c}_{11} = 1$, i.e., via considering the covering matrix investigated in the previous subsection. The $\tilde{c}_{11} = 1$ choice is promising because the homoclinic saddle point P,

a fixed point of $\mathcal{H}$, is contained in $Q_1 \cap \mathcal{H}^2(Q_1)$. (A periodic point of period two is contained in $Q_3 \cap \mathcal{H}^2(Q_3)$, and a periodic point of period 4 is contained in $Q_4 \cap \mathcal{H}^2(Q_2)$ and in $Q_2 \cap \mathcal{H}^2(Q_4)$ each.) This was the line of argumentation that motivated Subsection 4.2 above.

Remaining at two quadrangles, we firmly hope our optimization procedure leads to a proof of the following conjecture generalizing of what is presented in Figure 5 for $\ell = 2$.

Conjecture 1 *The full shift on two symbols embeds in the ℓ–s iterate of $\mathcal{H}$ if and only if $\ell = 2, 4$ or $\ell \geq 6$.*

Abstract theory implies embeddability only for ℓ sufficiently large. The proof of the Conjecture begins by showing that the *sufficiently large* number equals 12. This is also computer-assisted, accompanying the abstract argumentation in [14]. Cases $\ell = 2, 4, 6, 7, 8, 9, 10, 11$ had to be checked individually. The remaining cases $\ell = 1, 3, 5$ are excluded by Szymczak [15].

5. Conclusion

In the last paper of Galias [8] devoted to the subject, he presents the configuration of 29 polygons leading to the rigorous entropy estimate

$$h(\mathcal{H}) > 0.430\,.$$

This is quite near to the generally approved value of $h(\mathcal{H}) = 0.465\ldots$ conjectured on the basis of the number of periodic points of low periods. All the 29 polygons are narrow quadrangles (or quadrangles with some of the vertices "chopped off") situated along the unstable manifold of the homoclinic saddle. They were found by hand, based on earlier computer search for periodic points of low periods. If a global search is concentrated only on finding 29 segments of the unstable manifold, we need 58 parameters. This is too much for our optimization procedure which can hardly work with more than 20 parameters. So we are not able to reproduce Galias's best lower bound by our method at the moment. It remains open if a bootstrap application of our optimization procedure, keeping the number of parameters under say 10 at each step of the gradual improvements by the consecutive local searches, can reach a better entropy estimate.

Nevertheless, we think that in the near future automatic methods for rigorous chaos detection – including those that lead to rigorous estimates for quantitative chaos indicators – will be competitive to those requiring repeated human interference.

6. Acknowledgements

This work has been partially supported by the Hungarian National Science Foundation Grant OTKA No. T048377, T046822, and T049819. Barnabás Garay is also affiliated to Analogic and Neural Computing Laboratory, Computer and Automation Institute, H-1111 Budapest. The authors are grateful to the two anonymous referees for their suggestions that improved the presentation of the paper considerably.

References

[1] G. Alefeld and G. Mayer. Interval analysis: Theory and applications. *J. Comput. Appl. Math.*, 121:421–464, 2000.

[2] B. Bánhelyi, T. Csendes, and B. M. Garay. Optimization and the Miranda approach in detecting horseshoe-type chaos by computer. *International Journal of Bifurcation and Chaos*, 2007. Accepted for publication, available at `www.inf.u-szeged.hu/∼csendes/publ.html`.

[3] T. Csendes. Nonlinear parameter estimation by global optimization - efficiency and reliability. *Acta Cybernetica*, 8:361–370, 1988.

[4] T. Csendes, B. M. Garay, and B. Bánhelyi. A verified optimization technique to locate chaotic regions of a Hénon system. *Journal of Global Optimization*, 35:145–160, 2006.

[5] M. Dellnitz and O. Junge. *Handbook of Dynamical Systems*, volume 3, chapter Set oriented numerical methods for dynamical systems. North-Holland, 2002.

[6] Z. Galias. Rigorous numerical studies of the existence of periodic orbits for the Hénon map. *J. Universal Comp. Sci*, 4:114–124, 1998.

[7] Z. Galias. Interval methods for rigorous investigations of periodic orbits. *Int. J. Bifurcation and Chaos*, 11:2427–2450, 2001.

[8] Z. Galias. Obtaining rigorous bounds for topological entropy for discrete time dynamical systems. In *Proc. Internat. Symposium on Nonlinear Theory and its Applications*, pages 619–622, 2002.

[9] Z. Galias and P. Zgliczynski. Abundance of homoclinic and heteroclinic connections and rigorous bounds for the topological entropy of the Hénon map. *Nonlinearity*, 14:903–932, 2001.

[10] M. Gidea and P. Zgliczynski. Covering relations for multidimensional dynamical systems. *J. Differ. Eq.*, 202:32–58, 2004.

[11] K. Mischaikow and M. Mrozek. Chaos in the Lorenz equations: a computer-assisted proof. *Bull. Amer. Math. Soc.*, 32:66–72, 1995.

[12] M. Pireddu and F. Zanolin. Fixed points for dissipative–repulsive systems and topological dynamics of mappings defined on n–dimensional cells. *Advanced Nonlinear Studies*, 5:411–440, 2005.

[13] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Ellis Horwood, 1988.

[14] C. Robinson. *Dynamical Systems. Stability, Symbolic Dynamics, and Chaos*. RCR Press, 1999.

[15] A. Szymczak. A combinatorial procedure for finding isolated neighbourhoods and index pairs. *Proc. R. Soc. Edinb. A*, 127:1075–1088, 1997.

[16] P. Zgliczynski. Computer assisted proof of chaos in the Rössler equations and in the Hénon map. *Nonlinearity*, 10:243–252, 1997.

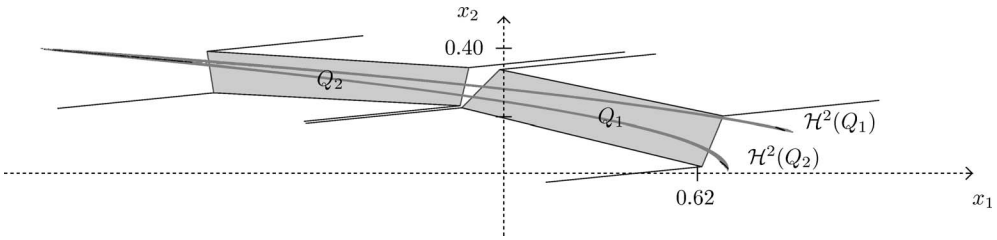


Figure 5. Two quadrangles found by computer

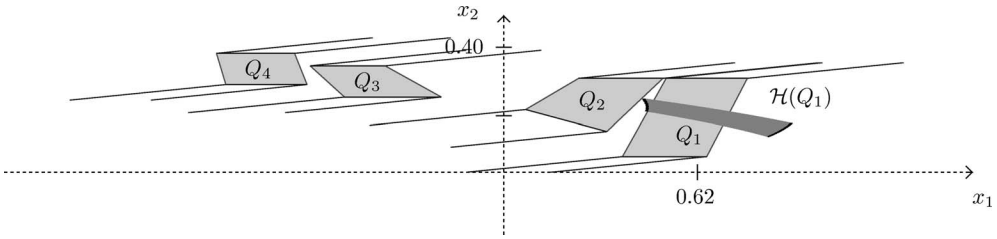


Figure 6. Four quadrangles found by computer

[17] P. Zgliczynski. Computer assisted proof of the horseshoe dynamics in the Hénon map. *Random Comput. Dynam.*, 5:1–17, 1997.

Nonlinear Adaptive Control of an Uncertain Wastewater Treatment Model

Neli Dimitrova, Mikhail Krastanov
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8, 1113 Sofia, Bulgaria
nelid@bio.bas.bg, krast@math.bas.bg

Abstract

A nonlinear model of an anaerobic digester wastewater treatment process is considered. Assuming that the model parameters are unknown but bounded, the asymptotic stabilizability of the control system is studied and a new adaptive stabilizing feedback control law is proposed. Computer simulations are also presented to illustrate the theoretical results.

1. Introduction

The interest to biological wastewater treatment (WWT) processes has recently highly increased due to the strong necessity of keeping the quantity of organic matter in industrial and urban effluents up to a critical level. This necessity has led to the development of adequate mathematical models and to the application of various techniques for monitoring, optimization and control of the processes.

The biological WWT processes have very specific features, induced by the presence of living organisms, whose activity depends on their physiological features as well as on the organisms environment.This is especially the case of mixed cultures where the interaction between the species is fairly complex. In the most cases the reproducibility of laboratory and practical experiments is not guaranteed, hence the experimental data are noisy and the reason for the noise is difficult to specify. As a consequence, it is impossible to determine the exact values of the parameters in the corresponding mathematical model. Additionally, the influent substrate concentrations of the WWT plants can also not remain constant over a long time period, because the quality of the wastewater depends e. g. on the raw materials of the production process, weather and seasonal changes [1]. In conclusion, WWT models are described by highly uncertain nonlinear dynamic systems and pose a challenge to the development of suitable control techniques.

One of the approaches for control design is the so called local approach. It is based on linearization of the system around a desired operating point [7]. This approach is known to be inefficient due to the strong nonlinearity of the models. Another strategy, the so called linearizing controller, is proposed in [2]; it transforms the nonlinear system into a linear one and uses perfect model knowledge. Global control strategies assuming model uncertainty are proposed for example in [13]. The latter are based on new techniques for monitoring and control (cf. for example [1], [2], [5], [8], [12] and the references there).

To the authors' knowledge, the application of adaptive feedback control laws seems to be a very appropriate tool for asymptotic stabilization in the case of model uncertainties. Such a feedback is designed in [11] for a simple model, the so-called "single substrate/single biomass" model of a continuous stirred tank bioreactor with unknown kinetics. The feedback stabilizes asymptotically the system to a previously chosen operating point and depends on both state variables – the substrate and the biomass concentrations. The real-life experiments show the practical relevance of the proposed approach.

In this paper, we also study the asymptotic stabilizability problem by using adaptive stabilizing feedback control laws. We consider a four-dimensional nonlinear model of an anaerobic digester wastewater treatment process, based on two microbial populations and two organic substrates. This model has been proposed in [9] as a theoretical base for general purpose studies. Assuming that all model parameters are unknown but bounded, a new adaptive stabilizing feedback law is proposed. Then an explicit Lyapunov function is constructed to prove that the closed-loop system is asymptotically stable in a neighborhood of an equilibrium point, which corresponds to a previously chosen operating point. Since the biomass concentrations are usually not online measurable, in order to realize practically the proposed feedback, the so called software sensors (observers) can be used [2], [3], which are robust with respect to disturbances and uncertainties [1], [8].

The paper is organized as follows. Section 2 presents the nonlinear WWT model. Steady states analysis involving model uncertainty is given in Section 3. A nonlinear adap-

Table 1. Definition of the model variables and parameters

t	time [day]
s_1	concentration of the organic substrate, characterized by the chemical oxygen demand (COD) [g/l]
s_2	concentration of volatile fatty acids (VFA) [mmol/l]
x_1	concentration of acidogenic bacteria [g/l]
x_2	concentration of methanogenic bacteria [g/l]
u	dilution rate [day^{-1}]
s_1^i	influent concentration of s_1 [g/l]
s_2^i	influent concentration of s_2 [mmol/l]
k_1	yield coefficient for COD degradation [g COD/g x_1]
k_2	yield coefficient for VFA production [mmol VFA/g x_1]
k_3	yield coefficient for VFA consumption [mmol VFA/g x_2]
$\mu_{\max}$	maximum acidogenic biomass growth rate [day^{-1}]
μ_0	maximum methanogenic biomass growth rate [day^{-1}]
k_{s_1}	saturation parameter associated with s_1 [g COD/l]
k_{s_2}	saturation parameter associated with s_2 [(mmol VFA/l)$^{1/2}$]
k_I	inhibition constant associated with s_2 [mmol VFA/l]
α	proportion of dilution rate reflecting process heterogeneity

tive feedback is proposed in Section 4 and a theorem on the asymptotic stabilizability of the control system is proven. Computer simulations in *Maple* demonstrating the theoretical results and the robustness of the proposed feedback are performed in Section 5.

2. General model description

We consider a model of an anaerobic digestion process, based on two microbial populations and two substrates, and described by the following system of nonlinear ordinary differential equations

$$\frac{ds_1}{dt} = u(s_1^i - s_1) - k_1\mu_1 x_1 \quad (1)$$

$$\frac{dx_1}{dt} = (\mu_1 - \alpha u)x_1 \quad (2)$$

$$\frac{ds_2}{dt} = u(s_2^i - s_2) + k_2\mu_1 x_1 - k_3\mu_2 x_2 \quad (3)$$

$$\frac{dx_2}{dt} = (\mu_2 - \alpha u)x_2. \quad (4)$$

The state variables s_1, s_2 and x_1, x_2 denote substrate and biomass concentrations, respectively: s_1 represents the organic substrate, characterized by its chemical oxygen demand (COD), s_2 denotes the volatile fatty acids (VFA), x_1 and x_2 are the acidogenic and methanogenic bacteria [1], [3], [4], [9], [14]. The definition of the model parameters is given in Table 1.

The parameter $\alpha \in [0, 1]$ represents the proportion of bacteria that are affected by the dilution; $\alpha = 0$ and $\alpha = 1$ correspond to an ideal fixed bed reactor and to an ideal continuous stirred tank reactor, respectively (cf. for example [1]). The dilution rate u is considered as a control variable which takes its values in a compact interval $\mathcal{U}$ of nonnegative real numbers.

The specific growth rate functions $\mu_1 = \mu_1(s_1)$ and $\mu_2 = \mu_2(s_2)$ are assumed to be of Monod and Haldane type, respectively: μ_1 and μ_2 are defined for $s_1, s_2 \in [0, +\infty)$, $\mu_j(0) = 0$ and $\mu_j(s_j) > 0$ whenever $s_j > 0$, $j = 1, 2$. Both functions are continuously differentiable and bounded, and

$\mu_1(s_1)$ is nondecreasing,

$\mu_1(s_1) \le \mu_{\max}$ for all $s_1 > 0$;

$\mu_2(s_2)$ has a maximum at $\tilde{s}_2 > 0$,

$\lim_{s_2 \to +\infty} \mu_2(s_2) = 0$,

$\mu_2(s_2)$ is strongly concave for $s_2 < \tilde{s}_2$.

As mentioned above, this model is proposed in [9] to describe the basic biological transformations in the reactor. In practice, for more accurate description of the process, this basic model can be extended by additional differential equations, see e. g. [1], [3], [4], [14].

3. Steady states analysis

Denote by ω the vector of the exact values of the model parameters, i. e. $\omega = (\alpha, k_1, k_2, k_3)$, and let μ_1 and μ_2 be the specific biomass growth rates as defined above. Excluding the trivial solutions $s_1 = s_1^i$, $s_2 = s_2^i$, $x_1 = 0$ and $x_2 = 0$ (called wash-out states) the equilibrium points of the system can be parameterized on u and ω as follows:

$$s_1(u;\omega) = \mu_1^{-1}(\alpha u) \quad (5)$$

$$x_1(u;\omega) = \frac{s_1^i - s_1(u;\omega)}{\alpha k_1} \quad (6)$$

$$s_2(u;\omega) = \mu_2^{-1}(\alpha u) \quad (7)$$

$$x_2(u;\omega) = \frac{s_2^i - s_2(u;\omega) + k_2\alpha x_1(u;\omega)}{\alpha k_3}; \quad (8)$$

here u belongs to the interval $U(\omega) = (0, u_1(\omega)]$ with

$$u_1(\omega) = \frac{1}{\alpha} \min \left\{ \mu_2 \left(\tilde{s}_2 \right),\ \mu_1(s_1^i) \right\}.$$

In (7), we have chosen for $s_2(u;\omega)$ that solution of the equation $\mu_2(s_2) = \alpha u$, which lies to the left of the maximum point $\tilde{s}_2$ of μ_2, that is $s_2(u;\omega) < \tilde{s}_2$; the reason is that the value of the other root of $\mu_2(s_2) = \alpha u$ tends to $+\infty$ as $u \to 0$. For simplicity, we denote the steady state vector by

$$\zeta(u;\omega) = (s_1(u;\omega), x_1(u;\omega), s_2(u;\omega), x_2(u;\omega)).$$

As mentioned above, the exact values of the parameter vector ω and of the biomass growth rates μ_1 and μ_2 are not known. Practical experiments and parameter estimation results (cf. [1], [8], [11], [15]) give only bounds for these quantities. For that reason, our basic assumptions are:

Assumption A1. The model parameters α, k_1, k_2 and k_3 are not exactly known but bounded within compact real intervals $[\alpha]$, $[k_1]$, $[k_2]$ and $[k_3]$, respectively, i. e.

$$\begin{aligned} &[\alpha] = [\alpha^-, \alpha^+], \quad 0 < \alpha^- \le \alpha^+ \le 1; \\ &[k_j] = [k_j^-, k_j^+], \quad 0 < k_j^- \le k_j^+,\ j = 1, 2, 3. \end{aligned}$$

Denote by $[\omega]$ the corresponding vector with interval components,

$$[\omega] = ([\alpha], [k_1], [k_2], [k_3]).$$

Assumption A2. Instead of the exact specific growth rates μ_1 and μ_2, we know bounds for them, namely

$$\begin{aligned} [\mu_1](s_1) &= [\mu_1^-, \mu_1^+](s_1) = [\mu_1^-(s_1), \mu_1^+(s_1)], \\ [\mu_2](s_2) &= [\mu_2^-, \mu_2^+](s_2) = [\mu_2^-(s_2), \mu_2^+(s_2)]. \end{aligned}$$

For any fixed u let $\Omega(u)$ be the set of all steady states, when the model parameters vary in the corresponding intervals, that is

$$\Omega(u) = \{ \zeta(u;\tilde{\omega}) \mid \tilde{\omega} \in [\omega] \}.$$

It is easy to see that the set $\Omega(u)$ is defined for u belonging to the interval

$$\mathbf{U} := \bigcap_{\tilde{\omega} \in [\omega]} U(\tilde{\omega}) = (0, u_1^+],$$

where

$$u_1^+ = \frac{1}{\alpha^+} \min \left\{ \mu_2^-(\tilde{s}_2^-),\ \mu_1^-(s_1^i) \right\}$$

and $\tilde{s}_2^-$ is the point where the function $\mu_2^-(s_2)$ takes its maximum.

Assumption A3. Let the set $\mathcal{U}$ of admissible values of the control u be a compact interval containing $\mathbf{U}$ in its interior, that is $\mathbf{U} \subseteq \mathit{int}\ \mathcal{U}$.

Let us fix an interval $[s_1^-, s_1^+]$ contained in the projection of the set $\bigcup_{u \in \mathbf{U}} \Omega(u)$ on the s_1-axis, that is

$$\begin{aligned} s_1^- &> \inf \left\{ (\mu_1^+)^{-1}(\alpha^- u) :\ u \in \mathbf{U} \right\}, \\ s_1^+ &\le \max \left\{ (\mu_1^-)^{-1}(\alpha^+ u) :\ u \in \mathbf{U} \right\}. \end{aligned} \tag{9}$$

Further, we fix an operating point s_1^* from the interval (s_1^-, s_1^+),

$$s_1^* \in (s_1^-, s_1^+).$$

Then there exists $u^* \in \mathbf{U}$, such that $s_1^* = s_1(u^*;\omega)$, where ω is the vector of the exact values of the unknown model parameters.

Assumption A4. In a neighborhood of s_1^*, the following inequality holds true

$$\frac{d}{ds_1} \left(\frac{\mu_1(s_1)}{\mu_1^+(s_1)} \right) > 0. \tag{10}$$

Assumption A4 is technical. It is not restrictive. As we shall see in Section 5, for the explicit Monod function μ_1 the inequality (10) is satisfied for all $s_1 > 0$.

In the next section we shall construct an adaptive feedback law, stabilizing asymptotically the system to the equilibrium point $\zeta^* := \zeta(u^*;\omega)$. This equilibrium point corresponds to the exact, but unknown parameter vector ω.

4. Adaptive asymptotic stabilization

We shall study the asymptotic stabilizability of the system (1)–(4) in a compact neighborhood Ω of the point ζ^*. First, we extend the system (1)–(4) by adding the following differential equation

$$\frac{d\beta}{dt} = -C\beta(1-\beta)\mu_1^+(s_1)(s_1 - s_1^*)x_1 \tag{11}$$

with the initial condition $\beta(0) \in (0,1)$. Here C is an arbitrary positive constant.

Define the following adaptive feedback control law

$$k(s_1, \beta) = \frac{1}{\alpha^-}\, \beta\, \mu_1^+(s_1). \tag{12}$$

The main result of the paper is the following

Theorem 1. Let the assumptions A1, A2, A3 and A4 hold true. Then there exists a compact neighborhood Ω of the point ζ^* such that for each point $\zeta \in \Omega$ the feedback $k(\cdot,\cdot)$ stabilizes asymptotically the control system (1)–(4) and (11) to (ζ^*, β^*) with $\beta^* = \dfrac{\alpha^-}{\alpha} \cdot \dfrac{\mu_1(s_1^*)}{\mu_1^+(s_1^*)}$.

Proof. Let us substitute in (1)–(4) and (11) the control input u by the feedback $k(\cdot,\cdot)$ and denote by $\Sigma(\omega)$ the obtained closed-loop system (with exact but unknown values for the model parameters). For convenience we set

$$\zeta = (s_1, x_1, s_2, x_2) \text{ and } \chi = (\zeta, \beta).$$

One can directly verify that the set Ω_0 with

$$\begin{aligned}\Omega_0 \;=\; \big\{\chi = (s_1, x_1, s_2, x_2, \beta) |\; & s_1^i > s_1 > 0,\\ & s_2 > 0,\ x_1 > 0,\ x_2 > 0,\ \beta \in (0,1)\big\}\end{aligned}$$

is strongly invariant with respect to (1)–(4) and (11) (cf. for example [6]). This means that every trajectory of $\Sigma(\omega)$ starting from a point $\chi \in \Omega_0$ remains in Ω_0. In particular, the coordinates of all points of this trajectory will never vanish.

The continuity of $k(\cdot,\cdot)$ implies the existence of some positive constants $b_{s_1}^-$, $b_{s_1}^+$ b_β^- and b_β^+ such that

$$b_{s_1}^- < s_1^* < b_{s_1}^+, \quad b_\beta^- < \beta^* < b_\beta^+,$$

and the values of the feedback $k(\cdot,\cdot)$ are admissible control values at each point of the set Ω_1, where

$$\begin{aligned}\Omega_1 \;=\; \{\chi = (s_1, x_1, s_2, x_2, \beta) \in \Omega_0 :\\ b_{s_1}^- \le s_1 \le b_{s_1}^+,\ b_\beta^- \le \beta \le b_\beta^+\},\end{aligned}$$

i. e. $k(s_1, \beta) \in \mathcal{U}$ for each point $\chi \in \Omega_1$.

Using the fact that

$$s_1^i = s_1^* + \alpha k_1 x_1^*$$

(see (6)), the first and the second equation of the closed-loop system $\Sigma(\omega)$ can be written as follows:

$$\begin{aligned}\frac{d}{dt}s_1 \;&=\; -k(s_1,\beta)(s_1 - s_1^* + \alpha k_1(x_1 - x_1^*))\\ &\quad - k_1(\mu_1(s_1) - \alpha k(s_1,\beta))x_1 \qquad (13)\\ \frac{d}{dt}x_1 \;&=\; (\mu_1(s_1) - \alpha k(s_1,\beta))x_1. \qquad (14)\end{aligned}$$

Consider the function

$$\begin{aligned}V_1(s_1, x_1, \beta) \;&=\; \frac{1}{\alpha(1-\alpha)}(s_1 - s_1^* + \alpha k_1(x_1 - x_1^*))^2\\ &\quad + k_1^2\,(x_1 - x_1^*)^2 + \frac{2k_1}{C\alpha^-}\int_{\beta^*}^{\beta}\frac{w - \beta^*}{w(1-w)}\,dw.\end{aligned}$$

Clearly, the values of this function are nonnegative. If we denote by $F_1(s_1, x_1, \beta)$ the right-hand side of (13)–(14) and (11), take into account the definition of the feedback control and apply the mean-value theorem, then it can be directly checked that

$$\begin{aligned}&\langle \text{grad } V_1(s_1, x_1, \beta), F_1(s_1, x_1, \beta)\rangle\\ &= -\frac{2\beta}{\alpha(1-\alpha)\alpha^-}\mu_1^+(s_1)\cdot(s_1 - s_1^*)^2\\ &\quad -\frac{2k_1x_1}{\alpha}\left(\mu_1'(\xi_1) - \frac{\alpha}{\alpha^-}\,\beta^*\mu_1^{+\prime}(\xi_2)\right)\cdot(s_1 - s_1^*)^2\\ &\quad -\frac{4k_1\beta}{(1-\alpha)\alpha^-}\,\mu_1^+(s_1)\cdot(s_1 - s_1^*)(x_1 - x_1^*)\\ &\quad -\frac{2\alpha k_1^2\beta}{(1-\alpha)\alpha^-}\,\mu_1^+(s_1)\cdot(x_1 - x_1^*)^2\end{aligned}$$

for each point $\chi = (s_1, x_1, s_2, x_2, \beta)$ of Ω_1, where ξ_i, $i = 1, 2$, are suitably chosen points between s_1 and s_1^*; the prime in μ_1 and μ_1^+ means $\dfrac{d}{ds_1}$.

The discriminant $D_1(s_1, x_1, \beta)$ of the last expression, considered as a quadratic function with respect to $s_1 - s_1^*$ and $x_1 - x_1^*$, is equal to

$$-\frac{4k_1^3\beta x_1}{(1-\alpha)\alpha^-}\,\mu_1^+(s_1)\cdot\left(\mu_1'(\xi_1) - \frac{\alpha}{\alpha^-}\,\beta^*\mu_1^{+\prime}(\xi_2)\right).$$

According to Assumption A4,

$$\mu_1'(s_1^*) - \frac{\alpha}{\alpha^-}\beta^*\mu_1^{+\prime}(s_1^*) = \mu_1^+(s_1^*)\cdot\frac{d}{ds_1}\left(\frac{\mu_1(s_1^*)}{\mu_1^+(s_1^*)}\right) > 0;$$

the last inequality implies that the inequality

$$\mu_1'(\xi_1) - \frac{\alpha}{\alpha^-}\beta^*\mu_1^{+\prime}(\xi_2) > 0$$

will also be valid in a sufficiently small neighborhood S_1 of s_1^*, and therefore $D_1(s_1, x_1, \beta) < 0$. Thus

$$\langle \text{grad } V_1(s_1, x_1, \beta), F_1(s_1, x_1, \beta)\rangle < 0$$

for each point $\chi = (s_1, x_1, s_2, x_2, \beta)$ from the set

$$\tilde{\Omega}_1 \setminus \{(\zeta^*, \beta) :\ \beta \in (0,1)\}$$

with

$$\tilde{\Omega}_1 = \{(\zeta, \beta) \in \Omega_1 :\ s_1 \in S_1\}.$$

Consider now the equations (3)–(4) with s_1, x_1 and u substituted by s_1^*, x_1^* and $u^* = \dfrac{1}{\alpha}\mu_2(s_2^*)$ respectively. By means of the relation

$$s_2^i = \alpha k_3 x_2^* + s_2^* - \alpha k_2 x_1^*$$

(see (8)), we obtain the following system:

$$\begin{aligned}\frac{d}{dt}s_2(t) \;&=\; -u^*(s_2 - s_2^* + \alpha k_3(x_2 - x_2^*))\\ &\quad - k_3(\mu_2(s_2) - \mu_2(s_2^*))x_2 \qquad (15)\\ \frac{d}{dt}x_2(t) \;&=\; (\mu_2(s_2) - \mu_2(s_2^*))x_2. \qquad (16)\end{aligned}$$

For each $\nu \in (0, \tilde{s}_2)$ define the set

$$\Omega_2^\nu := \{\chi = (s_1, x_1, s_2, x_2, \beta) \in \tilde{\Omega}_1 :\ s_2 \le \tilde{s}_2 - \nu\},$$

where $\tilde{s}_2$ is the point where $\mu_2(s_2)$ takes its maximum.

Consider the function

$$V_2(s_2,x_2) = (s_2-s_2^*+\alpha k_3(x_2-x_2^*))^2 + \alpha(1-\alpha)k_3^2(x_2-x_2^*)^2.$$

According to the mean-value theorem, there exists a point θ between s_2 and s_2^* such that

$$\begin{aligned}&\langle \text{grad } V_2(s_2,x_2), F_2(s_2,x_2)\rangle \\ &= -2(u^*+k_3(1-\alpha)\mu_2'(\theta)x_2)\cdot(s_2-s_2^*)^2 \\ &\quad -4k_3\mu_2(s_2^*)\cdot(x_2-x_2^*)(s_2-s_2^*) \\ &\quad -2\alpha k_3^2\mu_2(s_2^*)\cdot(x_2-x_2^*)^2,\end{aligned} \tag{17}$$

where by $F_2(s_2,x_2)$ we have denoted the right-hand side of (15)–(16) and $\mu_2' = \dfrac{d}{ds_2}\mu_2$.

Let $D_2(s_2,x_2)$ be the discriminant of the expression (17), considered as a quadratic function with respect to $s_2-s_2^*$ and $x_2-x_2^*$. The function $\mu_2(\cdot)$ is increasing and strictly concave on Ω_2^ν. Moreover, its derivative $\mu_2'(s_2)$ is a decreasing function which vanishes at the point $\tilde{s}_2$. Hence,

$$\mu_2'(\theta) > \mu_2'(\tilde{s}_2-\nu) > \mu_2'(\tilde{s}_2) = 0.$$

This and the definition of Ω_2^ν imply that on the set

$$\Omega_2^\nu \setminus \{\zeta^*,\beta) :\ \beta\in(0,1)\},$$

the following inequality holds true

$$\langle \text{grad } V_2(s_2,x_2), F_2(s_2,x_2)\rangle < 0,$$

because

$$D_2(s_2,x_2) < -4\alpha(1-\alpha)k_3^3\mu_2(s_2^*)\mu_2'(\tilde{s}_2-\nu)x_2 < 0.$$

For each $\varepsilon>0$ define further the function

$$V^\varepsilon(\chi) = V_1(s_1,x_1,\beta) + \varepsilon V_2(s_2,x_2). \tag{18}$$

Clearly, $V^\varepsilon(\cdot)$ is a smooth function which is nonnegative on the set Ω_2^ν and $V^\varepsilon(\chi^*)=0$ with $\chi^* := (\zeta^*,\beta^*)$.

Denote by $F(\cdot)$ the right-hand side of the closed-loop system $\Sigma(\omega)$. For an arbitrary and fixed point $\chi\in\Omega_2^\nu$, the scalar product

$$\langle \text{grad } V^\varepsilon(\chi), F(\chi)\rangle$$

is a quadratic function with respect to the variables $s_1-s_1^*$, $x_1-x_1^*$, $s_2-s_2^*$, $x_2-x_2^*$, whose coefficients depend on the components of χ. To check that it is negative definite, we calculate consecutively the leading principal minors of the corresponding symmetric matrix generated by the coefficients of the quadratic form:

$$\begin{aligned}\Delta_1^\varepsilon(\chi) &= -\frac{2}{\alpha(1-\alpha)}k(s_1,\beta) - \frac{2k_1}{\alpha}x_1\cdot \\ &\quad \cdot(\mu_1'(\xi_1) - \frac{\alpha}{\alpha^-}\beta^*\mu_1^{+\prime}(\xi_2)) < 0 \\ \Delta_2^\varepsilon(\chi) &= -D_1(s_1,x_1,\beta) > 0 \\ \Delta_3^\varepsilon(\chi) &< 2\varepsilon D_1(s_1,x_1,\beta) \\ &\quad \cdot(u^*+k_3x_2(1-\alpha)\mu_2'(\tilde{s}_2-\nu)) + o_1(\varepsilon) \\ \Delta_4^\varepsilon(\chi) &= \varepsilon^2 D_1(s_1,x_1,\beta)D_2(s_2,x_2) + o_2(\varepsilon^2).\end{aligned}$$

If $\varepsilon>0$ is sufficiently small, then the inequalities

$$\Delta_3^\varepsilon(\chi) < 0,\ \ \Delta_4^\varepsilon(\chi) > 0$$

hold true. Hence, for each $\chi\in\Omega_2^\nu\setminus\{(\zeta^*,\beta) :\ \beta\in(0,1)\}$,

$$\langle \text{grad } V^\varepsilon\chi), F(\chi)\rangle < 0.$$

Let us choose $\gamma>0$ as large as possible and such that

$$\Omega := \{\chi\in R^5 : V^\varepsilon(\chi)\le\gamma\ \}\subseteq\Omega_2^\nu.$$

Clearly, Ω is a compact neighborhood of the point $\chi^* = (\zeta^*,\beta^*)$. Applying the LaSalle's invariance principle (cf. for example [10]), it follows that every solution of the system (1)–(4), (11) is defined in the interval $[0,+\infty)$ and approaches the largest invariant set of (1)–(4), (11), which is contained in the set

$$\Omega\cap\left\{\chi=(\zeta,\beta)\in R^5 :\ \zeta=\zeta^*,\ \beta\in(0,1)\right\}.$$

It is easy to see that this invariant set consists of the single point (ζ^*,β^*). Therefore the feedback $k(\cdot,\cdot)$ stabilizes asymptotically the control system (1)–(4), (11) to the point (ζ^*,β^*) on the set Ω.

This completes the proof.

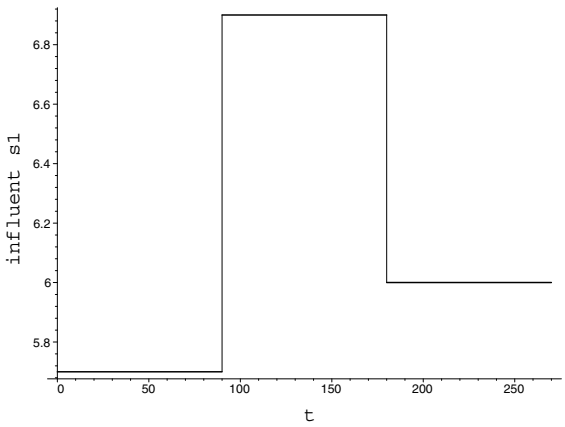


Figure 1. Plot of s_1^i.

5. Numerical simulation

We assume that the specific growth rates $\mu_1(s_1)$ and $\mu_2(s_2)$ are represented explicitly by the following Monod

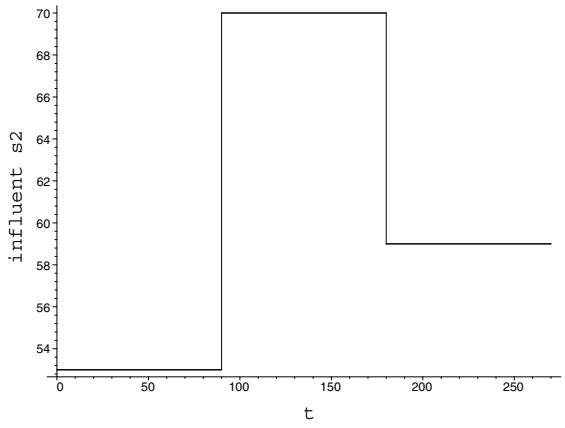


Figure 2. Plot of s_2^i.

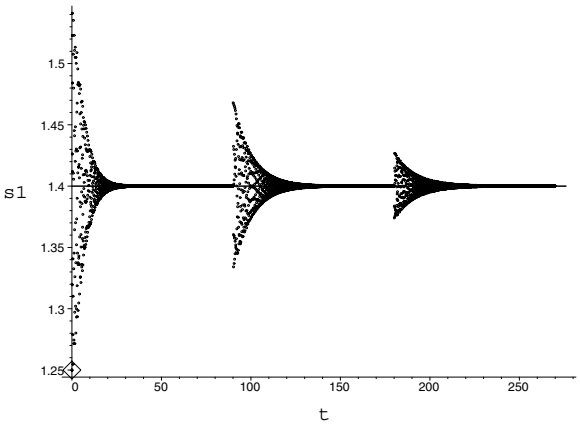


Figure 3. Time evolution of $s_1(t)$. The horizontal line segment goes through s_1^*.

and Haldane model functions:

$$\begin{aligned} \mu_1(s_1) &= \frac{\mu_{\max} s_1}{k_{s_1} + s_1} \\ \mu_2(s_2) &= \frac{\mu_0 s_2}{k_{s_2} + s_2 + \left(\dfrac{s_2}{k_I}\right)^2}. \end{aligned}$$

Let the values of the model parameters (see Table 1) $\mu_{\max}$, k_{s_1}, μ_0, k_{s_2} and k_I be enclosed by the intervals $[\mu_{\max}]$, $[k_{s_1}]$, $[\mu_0]$, $[k_{s_2}]$ and $[k_I]$, respectively. Then

$$\begin{aligned} [\mu_1](s_1) &= \left[\frac{\mu_{\max}^- s_1}{k_{s_1}^+ + s_1}, \frac{\mu_{\max}^+ s_1}{k_{s_1}^- + s_1}\right] \\ [\mu_2](s_2) &= \left[\frac{\mu_0^- s_2}{k_{s_2}^+ + s_2 + \left(\dfrac{s_2}{k_I^-}\right)^2}, \frac{\mu_0^+ s_2}{k_{s_2}^- + s_2 + \left(\dfrac{s_2}{k_I^+}\right)^2}\right]. \end{aligned}$$

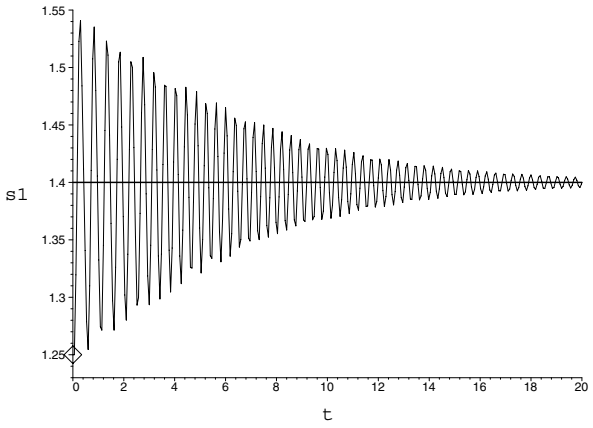


Figure 4. Fragment of $s_1(t)$ for $t \in [0, 20]$. The horizontal line segment goes through s_1^*.

As mentioned before, Assumption A4 is satisfied for all $s_1 > 0$ because

$$\frac{d}{ds_1}\left(\frac{\mu_1(s_1)}{\mu_1^+(s_1)}\right) = \frac{\mu_{\max}}{\mu_{\max}^+} \cdot \frac{k_{s_1} - k_{s_1}^-}{(k_{s_1} + s_1)^2} > 0.$$

Let $\omega = (\alpha, k_1, k_2, k_3, k_I, k_{s_1}, k_{s_2}, \mu_0, \mu_{\max})$ be now the vector of the exact values of the model parameters, and $[\omega]$ be the enclosing interval vector. The exact equilibrium points $s_1(u;\omega)$ and $s_2(u;\omega)$ from (5) and (7) can be explicitly computed

$$\begin{aligned} s_1(u;\omega) &= \frac{\alpha u k_{s_1}}{\mu_{\max} - \alpha u} \\ s_2(u;\omega) &= \frac{2\alpha u k_{s_2}}{\mu_0 - \alpha u + \sqrt{\Delta(u;\omega)}} \\ \Delta(u;\omega) &= \alpha^2\left(1 - 4\frac{k_{s_2}}{k_I^2}\right)u^2 - 2\alpha\mu_0 u + \mu_0^2 \end{aligned}$$

with $u \in U(\omega) = (0, u_1(\omega)]$,

$$u_1(\omega) = \frac{1}{\alpha}\min\left\{\mu_2(\tilde{s}_2),\ \mu_1(s_1^i)\right\}, \quad \tilde{s}_2 = k_I\sqrt{k_{s_2}}.$$

When ω varies in $[\omega]$, the admissible interval $\mathbf{U}$ is presented by $\mathbf{U} = (0, u_1^+]$ with

$$u_1^+ = \frac{1}{\alpha^+}\min\left\{\frac{\mu_0^-}{1 + 2\sqrt{k_{s_1}^+/k_I^-}}, \frac{\mu_{\max}^- s_1^i}{k_{s_1}^+ + s_1^i}\right\}.$$

Further, the interval $[s_1^-, s_2^+]$ (which is included in the projection of the set $\bigcup_{u\in\mathbf{U}} \Omega(u)$ on the s_1-axis) is given by

$$\begin{aligned} s_1^- &> \inf\left\{\frac{\alpha^- k_{s_1}^- u}{\mu_{\max}^+ - \alpha^- u} : u \in \mathbf{U}\right\} = 0 \\ s_1^+ &\leq \max\left\{\frac{\alpha^+ k_{s_1}^+ u}{\mu_{\max}^- - \alpha^+ u} : u \in \mathbf{U}\right\}. \end{aligned}$$

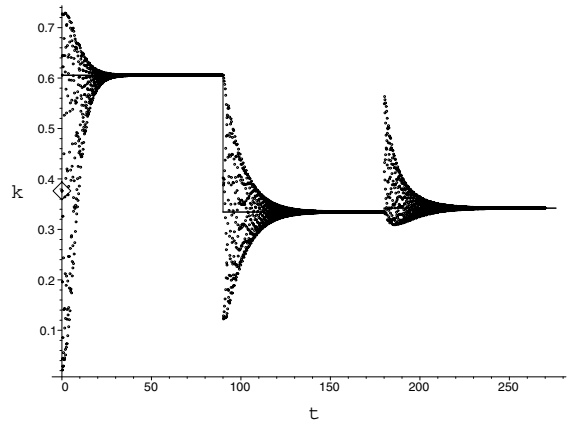


Figure 5. Time evolution of $k(t)$. The horizontal line segments go through the three different values for u^*.

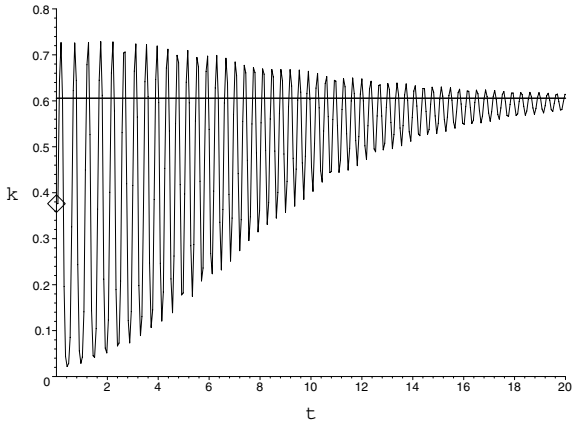


Figure 6. Fragment of $k(t)$ for $t \in [0, 20]$. The horizontal line segment goes through u^*.

Consider the following intervals for the model parameters:

$$
\begin{array}{llllll}
[\alpha] & = & [0.3,\ 0.6] & & & \\
[k_1] & = & [9.5,\ 11.5] & [k_2] & = & [27.6,\ 29.6] \\
[k_3] & = & [1064,\ 1084] & [k_{s_1}] & = & [6.5,\ 7.9] \\
[k_{s_2}] & = & [8.28,\ 10.28] & [k_I] & = & [15,\ 17] \\
[\mu_0] & = & [0.64,\ 0.84] & [\mu_{\max}] & = & [1,\ 1.4].
\end{array}
$$

These intervals are chosen to enclose the experimentally validated values of the parameters given in [1].

In practice, the influent concentrations s_1^i and s_2^i are not exactly known. For the computer simulation we assume that they are piece-wise constant functions taking the following numerical values [1]:

$$
s_1^i \ = \ \begin{cases} 5.7, & 0 \leq \ t \ \leq t_1 \\ 6.9, & t_1 < \ t \ \leq t_2 \\ 6, & t_2 < \ t \ \leq t_3; \end{cases}
$$

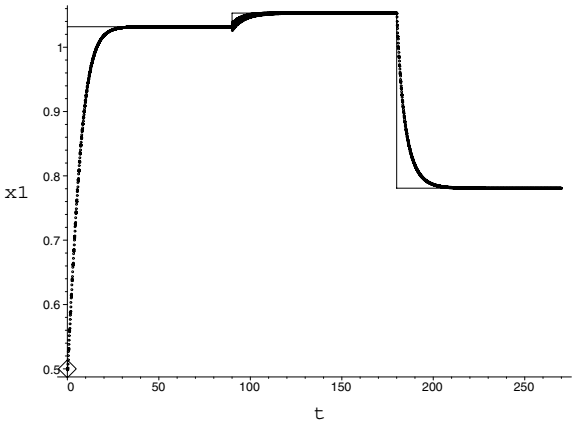


Figure 7. Time evolution of $x_1(t)$. The horizontal line segments go through the three different values of the equilibrium point x_1^*.

$$
s_2^i \ = \ \begin{cases} 53, & 0 \leq \ t \ \leq t_1 \\ 70, & t_1 < \ t \ \leq t_2 \\ 59, & t_2 < \ t \ \leq t_3. \end{cases}
$$

The graphics of s_1^i and s_2^i are presented on Figures 1 and 2 with $t_1 = 90$, $t_2 = 180$ and $t_3 = 270$.

Using the above numerical quantities, we compute

$$
\mathbf{U} \ \supseteq \ (0, 0.6985], \quad [s_1^-, s_1^+] \ \subseteq \ (0, 5.7].
$$

Further, we choose and fix the operating point

$$
s_1^* = 1.4.
$$

To simulate the theoretical results from the previous section and to demonstrate the robustness of the feedback with respect to the model uncertainties we proceed as follows. In the initial moment ($t_0 = 0$) we choose random values for the model parameters from the corresponding intervals and consider them as the exact vector ω. These values are kept constant for $t \in [t_0, t_1)$. At $t = t_1$, another set of random values for the model parameters is chosen to represent again the exact vector ω; these are kept constant for $t \in [t_1, t_2)$. Similarly, at $t = t_2$ a third set of randomly chosen parameter values is used to represent ω, which remain constant to the end of the computer experiment (at $t = t_3$). In all cases the closed loop system (1)–(4), (11) is solved numerically. Figures 3 to 10 visualize the numerical outputs.

Figure 3 presents the time profile of the state variable $s_1(t)$. The horizontal line segment corresponds to the previously chosen operating point s_1^*. Figure 4 represents a fragment of the solution $s_1(t)$ for $t \in [0, 20]$ to show the damped oscillations of the curve around the operating point. Similar behaviour of $s_1(t)$ is observed in small time intervals after t_1 and t_2 (when the model coefficients change

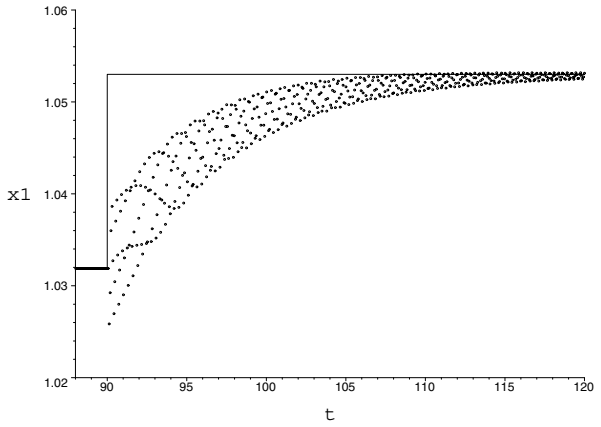


Figure 8. Fragment of $x_1(t)$ for $t \in [88, 120]$. The horizontal line segments go through the corresponding values of x_1^*.

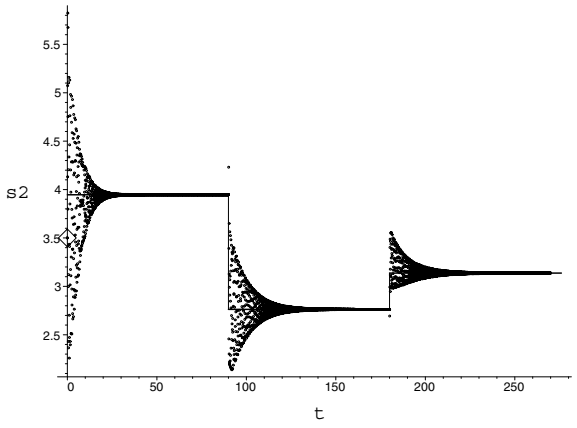


Figure 9. Time evolution of $s_2(t)$. The horizontal line segments go through the three different values of the equilibrium point s_2^*.

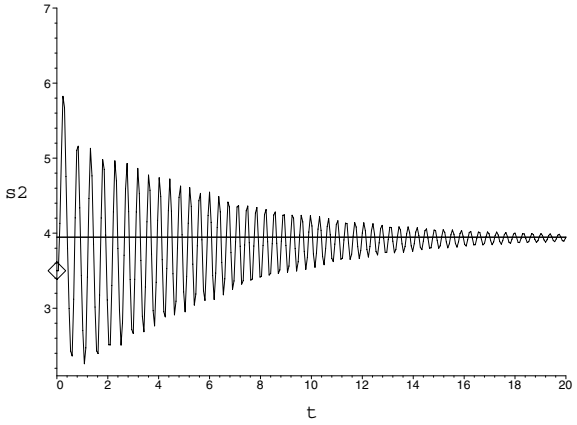


Figure 10. Fragment of $s_2(t)$ for $t \in [0, 20]$. The horizontal line segment goes through the corresponding value of s_2^*.

randomly in the corresponding intervals, and the influents s_1^i, s_2^i take different values).

Figures 5 and 6 show the time evolution of the feedback $k(\cdot, \cdot)$. The horizontal line segments go through the three different values of the point $u^* = \dfrac{1}{\alpha}\mu_1(s_1^*)$, corresponding to the three different sets of values for the parameters, s_1^i and s_2^i. Figure 6 shows the damped oscillation of the curve $k(t)$ around u^* for $t \in [0, 20]$.

Figure 7 visualizes the time evolution of the phase variable x_1. The horizontal line segments pass through the three different values of the equilibrium point x_1^*. A fragment of the curve $x_1(t)$ around $t = t_1$ is presented on Figure 8 to show its damped oscillations when the model parameters and s_1^i, s_2^i change. Figure 11 visualizes the solution $x_2(t)$; its behaviour around t_1 is similar to that of $x_1(t)$.

On Figures 9 and 10, the time profile of the state variable $s_2(t)$ is visualized. The horizontal line segments pass through to the three different values of the steady state s_2^*, which correspond to the three choices of the model parameters and s_1^i, s_2^i. Figure 7 presents a piece of the curve $s_2(t)$ for $t \in [0, 20]$, showing the damped oscillation of the solution around the corresponding steady state value. Similar behaviour of $s_2(t)$ is observed in small time intervals after t_2 and t_3 (where changes in the model parameters and in the influent concentrations occur).

On Figures 3, 4, 7 and 9 to 11, the symbol diamond $\diamond$ denotes the corresponding component of the initial point at $t_0 = 0$. The same symbol on Figures 5 and 6 denotes the value of the feedback $k(\cdot, \cdot)$ at the initial point.

Varying the influent concentrations s_1^i, s_2^i and the model coefficients simultaneously at the same time moments t_1 and t_2 is not restrictive. It is made for better quality and clarity of the plots. All computer experiments confirm the reliability of the proposed nonlinear control feedback: the closed-loop system stabilizes to the corresponding equilibrium point in practically reasonable time period.

6. Conclusion

The interest to biological wastewater treatment (WWT) processes has recently highly increased due to the strong necessity of keeping the quantity of organic matter in industrial and urban effluents up to a critical level. This necessity has led to the development of adequate mathematical models and to the application of various techniques for monitoring, optimization and control of the processes. The present paper is devoted to the design of an adaptive

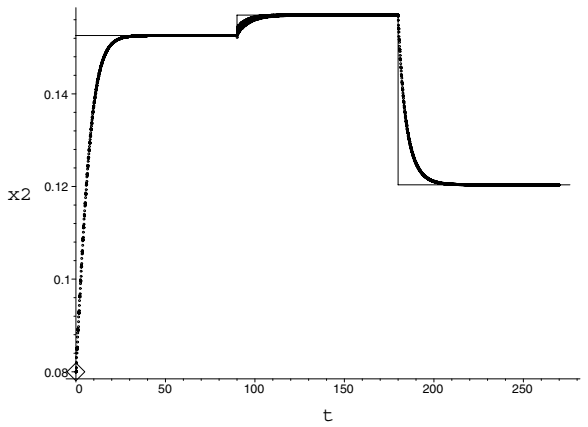


Figure 11. Time evolution of $x_2(t)$. The horizontal line segments go through the three different values of the equilibrium point x_2^*.

stabilizing feedback control law. A four-dimensional nonlinear model of an anaerobic digester wastewater treatment process is studied. This model has been build and validated in [9] to serve as a theoretical base for general purpose studies. Assuming that all model parameters are unknown but bounded, the asymptotic stabilizability of the control system is studied and a new adaptive nonlinear feedback law is proposed. This adaptive feedback stabilizes asymptotically the closed-loop system towards an (unknown) equilibrium point ζ^* such that its projection on the s_1-axis is equal to a previously chosen operating point s_1^*. Practically, the operating point s_1^* is chosen according to given environmental rules for guaranteeing an admissible level of the quantity of the organic matter released in industrial and urban effluents. The robustness of the adaptive feedback is demonstrated by assuming step changes in the influent concentrations s_1^i and s_2^i. The theoretical results are illustrated numerically in the environment of the computer algebra system *Maple*.

Acknowledgements. The authors are grateful to the anonymous referees for the invaluable advices and comments.

References

[1] V. Alcaraz-González, J. Harmand, A. Rapaport, J. P. Steyer, C. Pelayo-Ortiz. Software sensors for highly uncertain WWTPs: a new apprach based on interval observers. *Water Research*, 36:2515–2524, 2002.

[2] G. Bastin, D. Dochain. *On-line Estimation and Adaptive Control of Bioreactors*. Elsevier, Amsterdam, 1990.

[3] O. Bernard, Z. Hadj-Sadok, D. Dochain. Advanced monitoring and control of anaerobic wastewater treatment plants: dynamic model development and identification. *Proc. Fifth IWA Intern. Symp.* WATERMATEX 2000, Gent, Belgium, 3.57–3.64, 2000.

[4] O. Bernard, Z. Hadj-Sadok, D. Dochain, A. Genovesi, J.-P. Steyer. Dynamical model development and parameter identification for an anaerobic wastewater treatment process, *Biotech. Bioeng.*, 75:424–438, 2001.

[5] C. Ciccarella, M. Dalla, A. Germani. A Luenberger-like observer for nonlinear systems. *Int. J. Control*, 57:536–556, 1993.

[6] F. Clarke, Yu. Ledyaev, R. Stern, P. Wolenski. *Nonsmooth Analysis and Control Theory*. Graduate Text in Mathematics, vol. 178, Springer, Berlin, 1998.

[7] E. Heinzle, I. J. Dunn, G. B. Ryhiner. Modelling and control for anaerobic wastewater treatment. *Advances in Biochemical Engineering and Biotechnology*, 48:79–114, 1993.

[8] J.-L. Gouzé, A. Rapaport, Z. Hadj-Sadok. Interval observers for uncertain biological system. *Ecological Modelling*, 133:45–56, 2000.

[9] F. Grognard, O. Bernard. Stability analysis of a wastewater treatment plant with saturated control. *Wat. Sci. Tech.*, 53:149–157, 2006.

[10] H. K. Khalil. *Nonlinear Systems*. Macmillan Publishing Company, New York, 1992.

[11] L. Maillert, O. Bernard, J.-P. Steyer. Nonlinear adaptive control for bioreactors with unknown kinetics. *Automatica*, 40:1379–1385, 2004.

[12] I. Petersen, A. Savkin. *Robust Kalman filtering for signals and systems with large uncertainties*. Birkhäuser, Boston, 1999.

[13] A. Rapaport, J. Harmand. Robust regulation of a class of partially observed nonlinear continuous bioreactors. *Jrn. of Process Control*, 12:291–302, 2002.

[14] O. Schoefs, D. Dochain, H. Fibrianto, J.-P. Steyer. Modelling and identification of a distributed-parameter system for an anaerobic wastewater treatment process. *Chemical Eng. Research and Design*, 81(A9):1279–1288, 2003.

[15] I. Simeonov. Modelling and control of anaerobic digestion of organic waste. *Chemical and Biochemical Engineering Q.*, 8:45–52, 1994.

Interval Fuzzy Rule-Based Hand Gesture Recognition

Benjamín R. Callejas Bedregal
Depto de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Campus Universitário, 59.072-970 Natal, Brazil
bedregal@dimap.ufrn.br

Graçaliz P. Dimuro, Antônio C. Rocha Costa
Programa de Pós-Graduação em Informática
Universidade Católica de Pelotas
Felix da Cunha 412, 96010-000 Pelotas, Brazil
{liz,rocha}@ucpel.tche.br

Abstract

Abstract. *This paper introduces an interval fuzzy rule-based method for the recognition of hand gestures acquired from a data glove, with an application to the recognition of hand gestures of the Brazilian Sign Language. To deal with the uncertainties in the data provided by the data glove, an approach based on interval fuzzy logic is used. The method uses the set of angles of finger joints and of separation between finger for the classification of hand configurations, and classifications of segments of hand gestures for recognizing gestures. The segmentation of gestures is based on the concept of monotonic gesture segment, sequences of hand configurations in which the variations of the angles of the finger joints have the same sign (non-increasing or non-decreasing), separated by reference configurations that mark the inflexion points in the sequence. Each gesture is characterized by its list of monotonic segments. The set of all lists of segments of a given set of gestures determines a set of finite automata able to recognize such gestures.*

1. Introduction

Sign languages are the gestural languages used by deaf people in their daily face-to-face communication. Differently to the problems found in the processing of oral languages used by hearing people, the visual-gestural nature of sign languages gives rise to many specific problems for their automated recognition. Also, as it happens with spoken languages, sign languages are not universal, since they vary a lot from country to country. Although there is an extensive literature about methods and systems for gesture recognition in general, and hand gesture recognition in particular (see Section 6), it is possible to observe that, in spite of the existence of many works in the recognition of (American, Chinese, Arabian, etc.) Sign Languages, the automatic recognition of the Brazilian Sign Language (LIBRAS) [10] has not been extensively studied [33].

In this paper, we propose an interval fuzzy rule-based method for the recognition of hand gestures acquired from a data glove, extending the work presented in [4] to deal not only with the uncertainties in the recognition process, which is provided by the fuzzy logic theory [42], but also with the imprecision of the data provided by the glove, which is treated by Interval Mathematics [28]. We apply the method to the recognition of hand gestures of LIBRAS.

The method uses the set of angles of finger joints and of the separation between fingers (given as intervals that enclose the uncertainties of the data obtained by the glove sensors) for the classification of hand configurations, and classifications of sequences of hand configurations for recognizing gestures. The segmentation of gestures is based on the concept of *monotonic gesture segment*, sequences of gestures in which the variations of the angles of the finger joints have the same sign (non-increasing or non-decreasing), separated by reference hand configurations that mark the inflexion points in the sequence. Each gesture is characterized by a list of monotonic segments, which determine a set of finite automata, which are able to recognize the gestures being considered.

The paper is organized as follows. Section 2 presents a basic overview of fuzzy systems. Some concepts related to Interval Mathematics are discussed in Sect. 3. Our interval fuzzy rule-based method for hand gesture recognition is introduced in Sect. 4. The case study is presented in Sect. 5. A comparison with related work is introduced in Sect. 6. The Conclusion is in Sect. 7.

2. Fuzzy systems

Fuzzy sets were introduced in 1965 by Zadeh [42] for representing vagueness in everyday life, providing an approximate and effective means for describing the characteristics of a system that is too complex or ill-defined to be described by precise mathematical statements. In a fuzzy approach the relationship between elements and sets follows a transition from membership to non membership that

is gradual rather than abrupt.

A fuzzy system implements a function (usually nonlinear) of n variables, given by a linguistic description of the relationship between those variables. Figure 1 illustrates the architecture of standard fuzzy systems. The *fuzzificator* computes the membership degrees of the crisp input values to the linguistic terms (fuzzy sets) associated to each input linguistic variable. The *rule base* contains the inference rules that associate linguistic terms of input linguistic variables to linguistic terms of output linguistic values. The *information manager* is responsible for searching in the rule base which rules are applicable for the current input. The *inference machine* determines the membership degrees of the output values in the output sets, by the application of the rules selected in the rule base. The *defuzzificator* gives a single output value as a function of the output values and their membership degrees to the output sets. Applications were found in control systems [12], decision making [11], expert systems [37], etc.

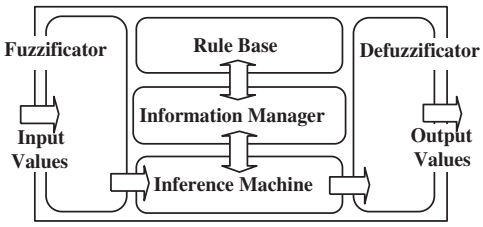


Figure 1. A standard fuzzy systems

However, many approximate methods do not produce a single final result, presenting several alternative solutions to a single problem (e.g., the different classes to which a given input may belong). Among them, several fuzzy rule-based methods for pattern recognition [27, 32], fuzzy relations, fuzzy clustering, fuzzy neural systems [25] were developed, with applications to signature verification [18], and face recognition [23], for example. Such methods consider an architecture like the one shown in Fig. 2. For applications to gesture recognition, see Sect. 6.

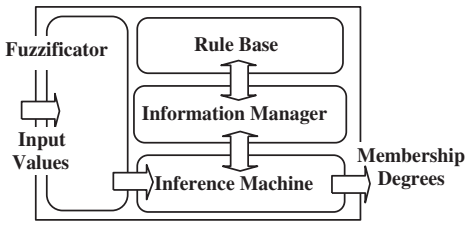


Figure 2. A fuzzy rule based system

Interval fuzzy rule-based systems consists of a generalization of such systems, by considering interval data type and interval membership degree values.

3. Interval Mathematics: some concepts

Interval Mathematics [28] is a mathematical theory that aims at the automatic and rigorous controlling of the errors that arise in numerical computations.

Any real number $x \in \mathbb{R}$ that is uncertain for some reason (e.g., if it is obtained by a measuring instrument with limited resolution) is represented by a real interval $X = [x_1; x_2]$, with $x_1, x_2 \in \mathbb{R}$ and $x_1 \leq x \leq x_2$. The set of intervals is denoted by $\mathbb{IR}$. x_1 and x_2 denote, respectively, the left and right endpoints of X.

A machine interval has floating point numbers as endpoints and outward roundings are used to guarantee that the resulting output interval of any computation process contains the actual result, with the range of the output interval being the indicative of the maximum error occurred in the process.

The arithmetical operations $*_{\mathbb{IR}} \in \{+, -, \times, \div\}$ are defined, for all $X, Y \in \mathbb{IR}$, as:

$$X *_{\mathbb{IR}} Y = \{x * y \mid x \in X, y \in Y\}. \quad (1)$$

For instance, if $X = [x_1; x_2]$ and $Y = [y_1; y_2]$, then

$$X - Y = [x_1 - y_2; x_2 - y_1] \text{ [28]}.$$

The *range* of a real function $f : \mathbb{R} \to \mathbb{R}$ over a real interval $X \in \mathbb{IR}$ is given by

$$\bar{f}(X) = \{f(x) \mid x \in X\}. \quad (2)$$

An *interval representation* of f is an interval function $F : \mathbb{IR} \to \mathbb{IR}$ such that for each $X \in \mathbb{IR}$, $f(x) \in F(x)$ whenever $x \in X$ [36]. Although there is not a unique interval representation for a given real function, it always hold that $\bar{f}(X) \subseteq F(X)$.

It is not always possible to represent interval functions in the cartesian plan. However, in some cases, an interval function $F(X)$ can be given by an interval of real functions

$$F(X) = [\inf\{f(x) \mid x \in X\}; \sup\{g(x) \mid x \in X\}], \quad (3)$$

where f and g are real functions such that $f \leq g$, which is denoted by $[f, g]$ [36].

In this work, we consider that the *membership functions* are interval functions F expressed as in (3), with $f = g$, denoted by $[f]$, which is the best interval representation of the function f, i.e., for any other interval representation F of f, $[f](X) \subseteq F(X)$, for any $X \in \mathbb{R}$ [36].

For the purpose of this work, the *sign* of a real interval $X = [x_1; x_2] \in \mathbb{IR}$ is defined as:

$$sign([x_1, x_2]) = \begin{cases} + & \text{if } x_1 \geq 0 \text{ and } x_2 > 0, \\ - & \text{if } x_1 < 0 \text{ and } x_2 \leq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The *absolute value* of a real interval $X = [x_1; x_2] \in \mathbb{IR}$ is given as:

$$| [x_1; x_2] | = max\{| x_1 |, | x_2 |\}. \quad (5)$$

4. The interval fuzzy rule-based method

We consider a hypothetical data glove with 19 sensors, as shown in Fig. 3. The fingers are labelled as: F1 (little finger), F2 (ring finger), F3 (middle finger), F4 (index finger) and F5 (thumb). The joints in the fingers are labelled as J1 (the knuckle), J2 and J3, for each finger. A separation between two fingers is labelled as Sij to indicate that it is a separation between the fingers Fi and Fj.

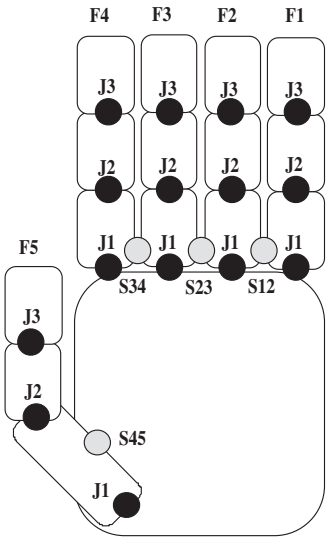


Figure 3. Localization of sensors in the data glove

Since any movement can be represented as a sequence of frames, a hand movement using a data glove is represented as a sequence of hand configurations, one for each discrete time instant. That is, at each time instant, the data glove sensors should provide the set of angles of joints and finger separation that characterizes a hand configuration. These angles are represented as *interval angles* $[x - \epsilon; x + \epsilon]$ that enclose the uncertainties in the processing, where x is the angle given by a sensor and $\epsilon > 0$, $\epsilon \in \mathbb{R}$, is the equipment tolerance, given by the manufacturer.

In order to simulate this data transfer, a random generator of hand configurations was implemented, generating at each instant one hand configuration represented by a tuple of interval angles corresponding to each sensor (see Fig. 3):

```
((F1J1,F1J2,F1J3),S12,(F2J1,F2J2,F2J3),S23,
 (F3J1,F3J2,F3J3),S34,(F4J1,F4J2,F4J3),S45,
 (F5J1,F5J2,F5J3))
```

Given a hand configuration c and a sensor s, denote the interval value of each sensor angle by $s(c)$, e.g., F1J1(c), S45(c) etc.

4.1 Fuzzification

When dealing with imprecise data represented by real intervals in $\mathbb{IR}$, one has the problem that $\mathbb{IR}$ does not present a natural total order. Considering that fuzzy systems usually work with totally ordered data, the designer often has difficulty to express the membership degrees as a function in the cartesian plan, for interval-valued data.

To solve this problem, interval fuzzy logic [13, 6, 3] was introduced to deal with interval membership degrees, i.e., subintervals of [0;1] that allow the expression of the uncertainty of the expert about the classification of the data in linguistic terms. In this work, the membership functions are given as explained in Sect. 3, that is, interval functions of type $[f]$, where f is a real membership function.

To each sensor corresponds a linguistic variable, whose values are linguistic terms representing typical angles of joints and separations.

For the joints in the fingers (linguistic variables F1J1, F1J2, F1J3 etc.) the linguistic terms are: STRAIGHT (St), CURVED (Cv) and BENT (Bt). Figures 4 and 5 show the (interval) fuzzification for those variables.

For the separations between fingers F1 and F2, F2 and F3, F4 and F5 (linguistic variable S12, S23, S45), the linguistic terms are: CLOSED (Cl), SEMI-OPEN (SOp) and OPEN (Op). For the separations between fingers F3 and F4 (linguistic variable S34), the linguistic terms are: CROSSED (Cr), CLOSED (Cl), SEMI-OPEN (SOp) and OPEN (Op). Figures 6 and 7 show the (interval) fuzzification for those variables.

4.2 The interval inference process

The more generic and accepted way to consider fuzzy generalization of classical connectives are based on triangular norms (t-norms, t-conorms, fuzzy negations and fuzzy implications (residuum)) [22]. In [6, 3] those concepts were defined in an interval approach.

In this paper, we use the generalization of the Gödel t-norm, given by

$$G([a, b], [c, d]) = [\min\{a, c\}; \min\{b, d\}]. \quad (6)$$

For example, consider the membership function for the joints of the index finger F4 (Fig. 5) and the rule

```
If F4J1 is STRAIGHT and
   F4J2 is CURVED and
   F4J3 is CURVED
Then F4 is StCvCv
```

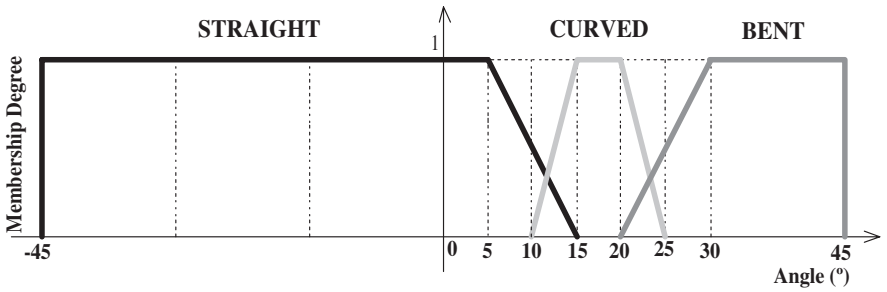


Figure 4. Fuzzification of the linguistic variable of the joint F5J2 in the thumb finger F5

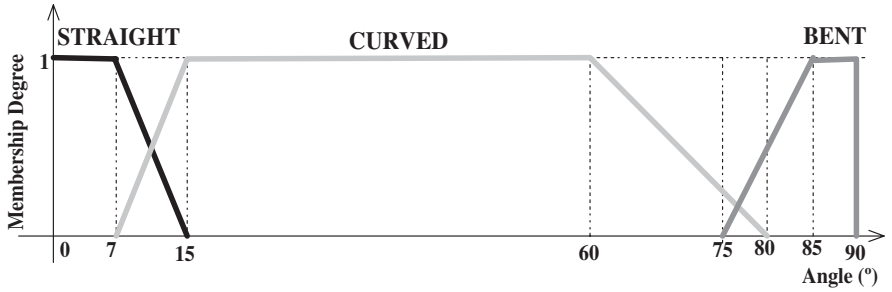


Figure 5. Fuzzification of the linguistic variables of remaining finger joints

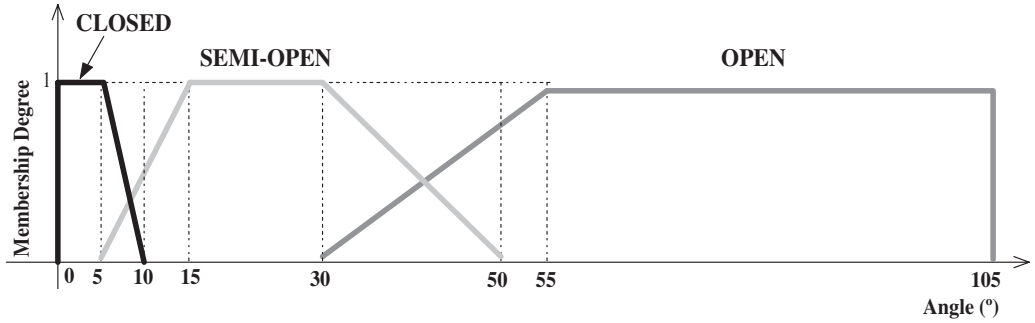


Figure 6. Fuzzification of the linguistic variable of the separation S45 between the index finger F4 and the thumb finger F5

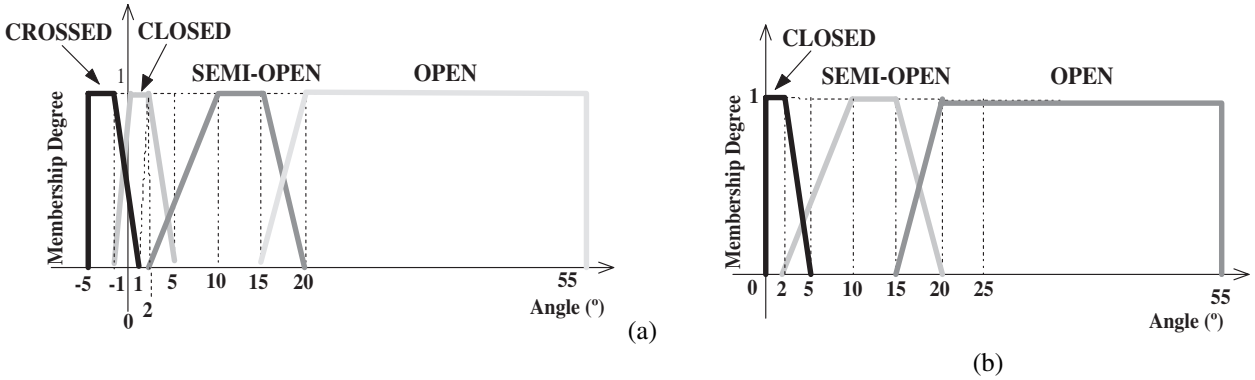


Figure 7. Fuzzification of the linguistic variables of the separations: (a) S34 between the middle finger F3 and the index finger F4, and (b) between remaining fingers

If the angles provided by the data glove for the joints J1, J2 and J3 are 7^o, 15^o and 13^o, respectively, and the tolerance is $\epsilon = 1^o$, then the interval membership degrees are:

$$\varphi_{F4J1}([6;8]) = \left[\frac{15-8}{8}; 1\right] = [0.875; 1],$$
$$\varphi_{F4J2}([14;16]) = \left[\frac{14-7}{8}; 1\right] = [0.875; 1],$$
$$\varphi_{F4J3}([12;14]) = \left[\frac{12-7}{8}; \frac{14-7}{8}\right] = [0.625; 0.875].$$

Figure 8 illustrates the processes for obtaining the interval membership degree of joint J3 in the index finger F4.

Then, by the interval t-norm G defined in (6), we obtain

$$G(G([0.875; 1], [0.875; 1]), [0.625; 0.875]) = [0.625; 0.875]$$

meaning that $F4$ is in StCvCv with interval degree [0.625;1]. If one uses the product interval t-norm, i.e.,

$$P([a; b], [c; d]) = [ac; bd], \tag{7}$$

the interval membership degree of finger F4 to StCvCv would be [0.546;0.875].

We observe that, in the fuzzification process, we considered only trapezoidal fuzzy sets and the interval Gödel t-norm, motivated just by simplicity.

4.3 The recognition process

The hand gesture recognition process is divided into four steps: (1) recognition of finger configurations; (2) recognition of hand configurations; (3) segmentation of the gesture in monotonic hand segments; (4) recognition of the sequence of monotonic hand segments.

For the Step 1 (*recognition of finger configurations*), 27 possible finger configurations are considered, for each finger. These configurations are codified in the format XYZ, where X, Y and Z are the values of the linguistic variables corresponding to the first joint J1, the second joint J2 and the third joint J3, respectively.

For example, StStSt indicates that the three joints are STRAIGHT, StCvCv indicates that the first joint is STRAIGHT whereas the others are CURVED etc.

The hand configuration is the main linguistic variable of the system, denoted by HC, whose linguistic terms are names of hand configurations, which names are application dependent. For instance, in Sect. 5, names of Brazilian Sign Language (LIBRAS) hand configurations (see Fig. 10) were used for such linguistic terms.

The 27 possible finger configurations determine 27 inference rules that calculate membership degree of each finger to each configuration. For example, see the rule for the index finger in the previous subsection.

Step 2 (*recognition of hand configurations*) determines the hand configuration, considering each finger configuration and separation between fingers. For example, the rule for the hand configuration [G] (Fig. 10) is:

```
If F1 is BtBtSt and S12 is Cl and
   F2 is BtBtSt and S23 is Cl and
   F3 is BtBtSt and S34 is Cl and
   F4 is StStSt and S45 is Cl and
   F5 is StStSt
Then HC is [G]
```

In Step 3 (*segmentation of the gesture in monotonic hand segments*), we divide each gesture in a sequence of k limit hand configurations $l_1, \ldots, l_k$, where l_1 is the initial gesture configuration and l_k is the terminal gesture configuration.

The limit configurations are such that, for each sensor s and $i = 1, \ldots, k-1$, it holds that:

(i) $\mid s(l_{i+1}) - s(l_i) \mid \leq 2\epsilon$, where the *absolute value* of an interval was defined in (5) and ϵ is the equipment tolerance.

(ii) For each c between l_i and l_{i+1},

$$sign(s(c) - s(l_i)) = sign(s(l_{i+1}) - s(l_i)).$$

(iii) For each c' after l_{i+1},

$$sign(s(c') - s(l_{i+1})) \neq sign(s(l_{i+1}) - s(l_i)),$$

where the *sign* of an interval was given in (3). A *sign* equal to 0 is compatible with both negative and positive signs.

The limit hand configurations are the points that divide the gesture into monotonic segments, that is, segments in which each sensor produces angle variations with constant (or null) sign. For each segment $l_i l_{i+1}$, l_i and l_{i+1} are its initial and terminal hand configurations, respectively.

The procedure for step 3 is the following. To find any monotonic segment $l_i l_{i+1}$, the next n configurations sent by the data glove after l_i are discarded, until a configuration c_{n+1}, such that

$$sign(s(c_{n+1}) - s(c_n)) \neq sign(s(c_n) - s(l_i))$$

(or, c_{n+1} is the last configuration of the gesture). Then, c_n (resp., c_{n+1}) is the terminal hand configuration l_{i+1} of the considered monotonic segment, and also coincides with the initial configuration of the next segment $l_{i+1} l_{i+2}$ (if there is one). The process starts with $l_i = l_1$, which is the initial gesture configuration, and is repeated until the end of the gesture, generating the list of k limit hand configurations.

In Step 4 (*recognition of the sequence of monotonic hand segments*), the recognition of each monotonic segment

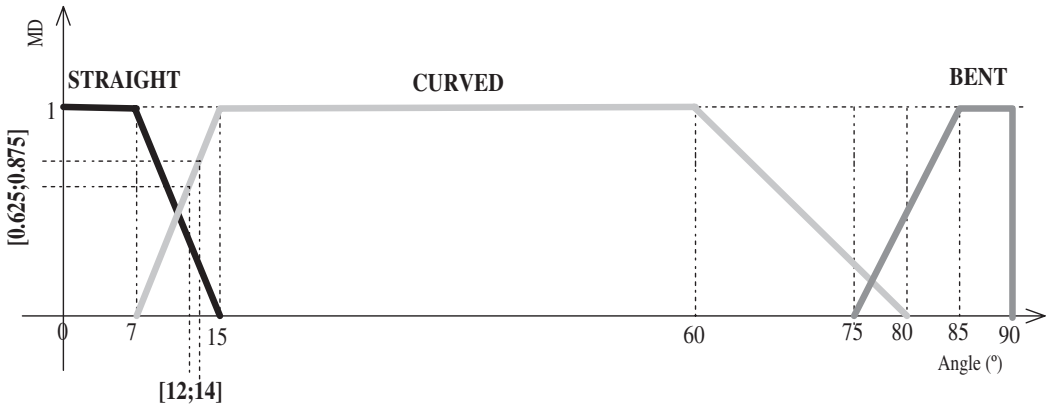


Figure 8. Interval membership degree (MD) of joint J3 in the index finger F4

$l_i l_{i+1}$ is performed using a list of reference hand configurations $r_1, r_2, \ldots, r_m$ that characterizes the segment, where r_1 and r_m are the initial and terminal hand configurations of the segment, respectively.

A monotonic segment is recognized by checking that if it contains its list of reference hand configurations. The process is equivalent to a recognition based on a linear finite automaton (shown in Fig. 9), where $l_i = r_1$ and $l_{i+1} = r_m$.

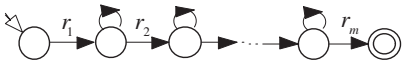


Figure 9. Automaton for the recognition of monotonic segments

5. Case study: hand gestures of LIBRAS

As any other sign language, LIBRAS (Brazilian Sign Language) is a natural language endowed with all the complexity normally found in the oral-auditive languages.

In the various works on automatic recognition of sign languages that have been developed along the years (see Sect. 6) the recognition of hand gestures has occupied a prominent place. To support that recognition process, a reference set of hand configurations is usually adopted, driven either from the linguistic literature on sign languages, or dynamically developed by the experimenters with an ad hoc purpose. For our purposes, we have chosen a standard set of hand configurations (some of them shown in Fig. 10), taken from the linguistic literature on LIBRAS [10].

Our method requires that each sign be thoroughly characterized in terms of its monotonic segments and the sequences of hand configurations that constitute such segments, and that the identification of the monotonic segments

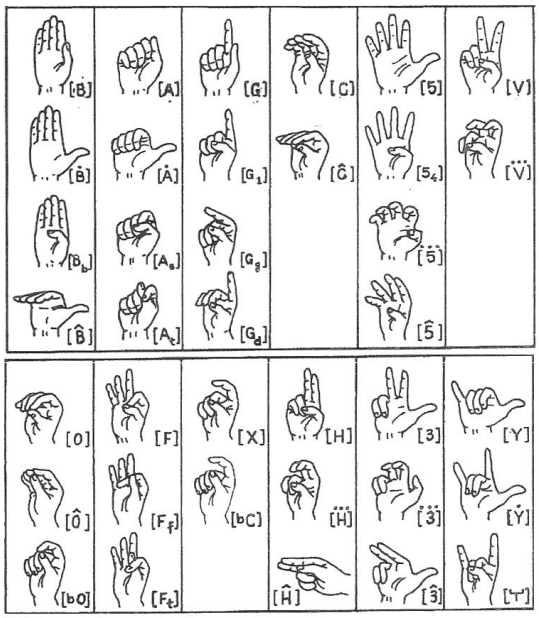


Figure 10. Some LIBRAS hand configurations [10]

and hand configurations be manually provided to the system. Although a capture device such as a data glove can be used to help to identify the typical values of the angles of the finger joints, the final decision about the form of the membership functions that characterize the linguistic terms has to be explicitly taken and manually transferred to the system. We illustrate the application of the method by the definition of the necessary parameters for the recognition of the hand gestures that constitute the signs CURIOUS and BIRD in LIBRAS.

CURIOUS is a sign performed with a single hand placed right in front of the dominant eye of the signer, with the palm up and hand pointing forward. The initial hand configuration is the one named [G1] in Fig. 10. The gesture consists of the monotonic movement necessary to perform the transition from [G1] to [X] and back to [G1] again, such movements been repeated a few times (usually two or three). A possible analysis of the gestures that constitute the sign CURIOUS is presented in Table 1.

To support the recognition of the monotonic segments of CURIOUS, we have chosen to use an intermediate hand configuration, [G1X], which does not belong to the reference set (Fig. 10) and whose characterization in terms of the set of membership functions for linguistic terms was defined in an ad hoc fashion, for the purpose of the recognition of CURIOUS. Together with [G1] and [X], it should be added to the list of hand configurations used by the system.

Table 1. Analysis of hand gestures

Steps	Sign CURIOUS	Sign BIRD
Initial Configuration	[G1]	[bC]
Monotonic Segment S1	[G1]-[G1X]-[X]	[bC]-[bO]
Monotonic Segment S2	[X]-[G1X]-[G1]	[bO]-[bC]
Recognition Automaton	Fig. 11(a)	Fig. 11(b)

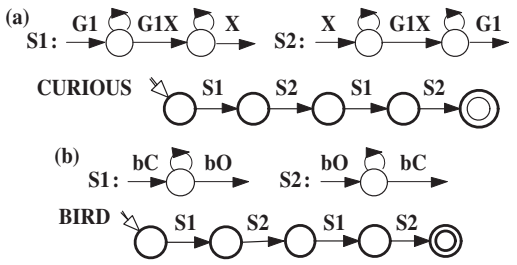


Figure 11. Recognition automata of hand gestures of the signs CURIOUS and BIRD

The sign BIRD is performed with one single hand, placed at the neutral signing space (mid arm distance from the signer), with the palm facing forward, fingers pointing up. A possible analysis of the hand gestures that constitute that sign is shown in the third column of Table 1. Observe that, since the difference in angles of finger joints between [bC] and [bO] is small enough, it was not necessary to require from the recognition process the identification of any special intermediate hand configuration.

The recognized signs are presented in a written form, using the HamNoSys [34] notation (a notation system developed for the linguistic study of sign languages) (see more details in [5]).

6. Considerations on the fuzzy approach for gesture recognition and related work

Fuzzy set theory is the oldest and most widely part of soft computing, which deals with the design of flexible information processing systems, providing soft decision by taking into account characteristics like tractability, robustness, low cost, etc., and have close resemblance to human decision making [27]. The significance of fuzzy set theory in the realm of pattern recognition was fully discussed in the literature (see, e.g, [27, 32]), where it was shown that it is adequately justified in the following cases: (i) representing linguistically phrased input features for processing; (ii) providing an estimate (representation) of missing information in terms of membership values; (iii) representing multiclass membership of ambiguous patterns and in generating rules and inferences in linguistic form; (iv) extracting ill-defined image regions, primitives, and properties and describing relations among them as fuzzy subsets.

Observe that fuzzy set theory provides a notion of embedding [27], since one finds a better solution to a crisp problem by looking in a large space at first, which has different (usually less) constraints, and, therefore, allows the algorithm more freedom to avoid errors forced by commission to definite answers in intermediate stages.

Applications of fuzzy pattern recognition and image processing have been reported in various domains [7, 30], like speech recognition, remotely sensed images, medical imagery, and atmospheric sciences and, in particular, in *gesture recognition* [8, 39]. Various hybrid approaches were also found for gesture recognition (see, e.g., [1, 9]), by combining the merits of individual techniques. For example, neuro-fuzzy models [1, 9] allow one to incorporate the generic advantages of artificial neural networks and fuzzy logic-like massive parallelism, robustness, learning, and handling of uncertainty and impreciseness, into the system. Also, it is found to perform better than either a neural network or a fuzzy system considered individually [43].[1] Other hybridizations of fuzzy sets have been investigated using other soft computing tools like genetic algorithms and rough sets [31], but not significantly for gesture recognition.

A popular fuzzy method for gesture recognition is the Bimber's algorithm [8], which is applicable to any object that can be tracked. This method stores each representation of a gesture as an analysis of 56 attributes. Each new input gesture is analyzed for the same 56 attributes and is compared to each stored representation to find the closest match. The main features of the method are its usability

[1] There exist non-fuzzy approaches for gesture recognition that aim at better performances, like, e.g., the work in [35, 29], based on Hidden Markov Models, or in [20], which uses Finite State Machines. However, one may claim that they do not present the flexibility of fuzzy techniques in handling uncertainties.

(e.g. with 2D or 3D input devices or in combination with finger status information, etc.), the minimum of information needed to recognize a gesture, and, consequently, the high speed of its scanning and comparison process, and the exceptionally low failure rate (less than 1%). The algorithm is very useful for gesture recognition based games. Some variants on Bimber's algorithm in an attempt to further improve its performance were present in [2]. The orientation-modified algorithm produced lower failure rates and, additionally, the speed at which a gesture is performed seems to have no bearing on whether or not it will be recognized.

Considering in particular sign languages, we observe that, unlike general gestures, sign language is highly structured so that it provides an appealing test bed for new ideas and algorithms before they are applied to gesture recognition. The recognition methods usually include rule-based matching, artificial neural networks [15], Hidden Markov Models [17, 24, 38, 40] and decision trees [14, 21, 26]. Most of these works are mainly concerned with the difficulty presented by a large vocabulary sign language and how to reduce the recognition time without a great loss of accuracy.

Fuzzy methods, however, are worried mainly with representing the uncertainties involved at every stage of the process, like, e.g., in [41], where the test of the system used just 16 words of Japanese Sign Language and the system correctly distinguished all words, or in [19], where the system was tested with 21 Auslan signs correctly recognized.

As those other fuzzy methods for sign recognition mentioned before, we are concerned in representing all the uncertainties related to recognition process, in other to increase the the quality of the results.[2] Then, the innovation of our method is the use of the association of fuzzy logic to interval mathematics to deal also with the imprecision of data provided by the data glove, which are then represented by real intervals. To avoid the difficulty in expressing the membership degrees for interval-valued data, we use interval membership degrees (as, e.g., in [13, 6, 3]), i.e., subintervals of [0;1] that allow the expression of the uncertainty of the expert about the classification of the data in linguistic terms. Also, we remark the contribution of this work to the automatic recognition of LIBRAS, which has been rarely studied [33]. In addition, we aim to meet the interest of linguistic studies in LIBRAS, since the system represents the recognized signs in the HamNoSys [34] linguistic notation (see [5], for details), which is not usual in other systems.

7. Conclusion and final remarks

This paper presented a fuzzy rule-based for the recognition of hand gestures. The method is highly dependent on a detailed previous analysis of the features of the gestures to be recognized, and on the manual transfer of the results of that analysis to the recognition system. This makes it suitable for the application to the recognition of hand gestures of sign languages, because of the extensive analysis that linguists that have already done of those languages.

Prototypes of a random gesture generator and of the gesture recognizer were implemented in the programming language Python, using the module *PyInterval* [16] for Interval Mathematics. The output of the recognition system was fed into a simple translator able to render the recognized hand gestures as they are annotated in the HamNoSys system. This initial experimentation indicated promising results in the direction of a system capable of providing a HamNoSys rendering of the recognized gestures.

Future work is concerned with the recognition of arm gestures, by including the analysis of the angles of arm joints, so recognition of more complete gestural features of signs can be achieved.

[2] At the moment, we are not interested in the evaluation of our method in relation to recognition time.

Acknowledgments

This work was partially supported by CNPq (Proc. 470871/2004-0, 470556/2004-8). The authors are very grateful to the referees for their valuable suggestions.

References

[1] O. Al-Jarrah and A. Halawani. Recognition of gestures in arabic sign language using neuro-fuzzy systems. *Artificial Intelligence*, 133:117–138, 2001.

[2] L. Anderson, J. Purdy, and W. Viant. Variations on a fuzzy logic gesture recognition algorithm. In *Proc. ACM SIGCHI Intl. Conf. Advances in computer entertainment technology*, pages 280–283, New York, 2004. ACM Press.

[3] B. C. Bedregal and A. Takahashi. Interval valued versions of T-conorms, fuzzy negations and fuzzy implications. In *IEEE Proc. Intl. Conf. Fuzzy Systems*, Vancouver, 2006.

[4] B. C. R. Bedregal, A. C. R. Costa, and G. P. Dimuro. Fuzzy rule-based hand gesture recognition. In M. Bramer, editor, *Artificial Intelligence in Theory And Practice*, number 217 in IFIP Series, pages 285–294, Boston, 2006. Springer.

[5] B. C. R. Bedregal, G. P. Dimuro, and A. C. R. Costa. Fuzzy rule-based hand gestures recognition for sign language processing. In S. O. Rezende and A. C. R. S. Silva Filho, editors, *Proc. of the Work. on Computacional Intelligence (WCI'06) in Intl. Joint Conf. 10th IBERAMIA, 18th SBIA, 9th SBRN*, Ribeirão Preto, 2006. ICMC-USP.

[6] B. C. R. Bedregal and A. Takahashi. The best interval representations of T-norms and automorphisms. *Fuzzy Sets and Systems*, 157(24):3220–3230, 2006.

[7] J. C. Bezdek and S. K. Pal, editors. *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. IEEE Press, New York, 1992.

[8] O. Bimber. Continuous 6DOF gesture recognition: A fuzzy-logic approach. In *Proc. of the 7th Intl. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, volume 1, pages 24–30, 1999.

[9] N. D. Binh and T. Ejima. Hand gesture recognition using fuzzy neural network. In *Proc. ICGST Intl. Conf. Graphics, Vision and Image Processing*, pages 1–6, Cairo, 2005.

[10] L. F. Brito. *Por uma Gramática de Línguas de Sinais*. Tempo Brasileiro, Rio de Janeiro, 1995. (in Portuguese).

[11] C. Carlsson and R. Fuller. *Fuzzy Reasoning in Decision Making and Optimization*. Springer, Heidelberg, 2002.

[12] G. Chen and T. T. Pham. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*. CRC Press, Boca Raton, 2001.

[13] D. Dubois and H. Prade. Interval-valued fuzzy sets, possibility theory and imprecise probability. In *Proc. Intl. Conf. Fuzzy Logic and Tech.*, pages 314–319, Barcelona, 2005.

[14] G. Fang, W. Gao, and D. Zhao. Large vocabulary sign language recognition based on fuzzy decision trees. *IEEE Trans. Systems, Man and Cybernetics*, 34:305–314, 2004.

[15] S. S. Fels and G. E. Hinton. Glove-talk: A neural network interface between a data-glove and a speech synthesizer. *IEEE Transactions on Neural Networks*, 4:1–8, 1993.

[16] P. S. Grigoletti, G. P. Dimuro, and L. V. Barboza. Módulo Python para matemática intervalar. In *Proc. 29th Congresso Nacional de Matemática Aplicada e Computacional*, Campinas, 2006. (in Portuguese, available at http://ppginf.ucpel.tche.br/gracaliz/papers).

[17] K. Grobel and M. Assan. Isolated sign language recognition using hidden markov models. In *Proc. Intl. Conf. System, Man and Cybernetics*, pages 162–167, 1997.

[18] M. Hanmandlu, M. H. M. Yusof, and V. K. Madasu. Off-line signature verification and forgery detection using fuzzy modeling. *Pattern Recognition*, 38(3):341–356, 2005.

[19] E. J. Holden and R. A. Owens. Visual sign language recognition. In *Theor. Found. Comp. Vision*, pages 270–288, 2000.

[20] P. Hong, M. Turk, and T. S. Huang. Gesture modeling and recognition using FSM. In *Proc. of IEEE Conf. Face and Gesture Recognition*, pages 410–415, Grenoble, 2000.

[21] M. W. Kadous. Machine recognition of auslan signs using powergloves: towards large-lexicon recognition of sign language. In *Proc. Work. Integration of Gesture in Language and Speech*, pages 165–174, 1996.

[22] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Press, Dordrecht, 2000.

[23] K. Kwak and W. Pedrycz. Face recognition using a fuzzy fisherface classifier. *Pattern Recognition*, 38(10):1717–1732, 2005.

[24] R. H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Proc. 3rd Intl. Conf. Automatic Face and Gesture Recognition*, pages 558–565, 1998.

[25] C. T. Lin and C. S. G. Lee. *Neural Fuzzy Systems: A neuro-fuzzy synergism to intelligent systems*. Prentice Hall, Upper Saddle River, 1996.

[26] H. Matsuo, S. Igi, S. Lu, Y. Nagashima, Y. Takata, and T. Teshima. The recognition algorithm with non-contact for japanese sign language using morphological analysis. In *Proc. of the Intl. Gesture Workshop*, pages 273–284, 1997.

[27] S. Mitra and S. K. Pal. Fuzzy sets in pattern recognition and machine intelligence. *Fuzzy Sets and Systems*, 156:381–386, 2005.

[28] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[29] X. Ou, X. Chen and J. Yang. Gesture recognition for remote collaborative physical tasks using tablet PCs. In *Proc. of the 9th IEEE Intl. Conf. on Computer Vision, Work. on Multimedia Tech. in E-Learning and Collaboration*, Nice, 2003.

[30] S. K. Pal. Fuzzy image processing and recognition: uncertainty handling and applications. *Intl. Journal Image Graphics*, 1:169–195, 2001.

[31] S. K. Pal and A. Skowron, editors. *Rough-Fuzzy Hybridization: A New Trend in Decision Making*. Springer, Heidelberg, 1999.

[32] W. Pedrycz. Fuzzy sets in pattern recognition: Accomplishments and challenges. *Fuzzy Sets and Systems*, 90:171–176, 1997.

[33] H. Pistori and J. J. Neto. An experiment on handshape sign recognition using adaptive technology: Preliminary results. In *Advances in Artificial Intelligence: Proc. 17th Braz. Symp. Artificial Intelligence, São Luis, 2004*, number 3171 in LNCS, pages 464–473, Berlin, 2004. Springer.

[34] S. Prillwitz, R. Leven, H. Zienert, T. Hanke, J. Henning, E. Richter, and J. Martin. *HamNoSys. V. 2.0; Hamburg Notation System for Sign Languages. An introductory guide*. Number 5 in Intl. Studies on Sign Language and Communication of the Deaf. Signum, Hamburg, 1989.

[35] G. Rigoll, A. Kosmala, and S. Eickeler. High perfomance real-time gesture recognition using Hidden Markov Models. In I. Wachsmuth and M. Frölich, editors, *Gesture and Sign Language in Human-Computer Interaction*, number 1371 in LNAI, pages 69–80, Berlin, 1998. Springer.

[36] R. H. N. Santiago, B. C. Bedregal, and B. M. Acióly. Formal aspects of correctness and optimality of interval computations. *Formal Aspects of Computing*, 18:231–243, 2006.

[37] W. Siler and J. J. Buckley. *Fuzzy Expert Systems and Fuzzy Reasoning*. John Wiley & Sons, New York, 2004.

[38] T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[39] M. Su. A fuzzy rule-based approach to spatio-temporal hand gesture recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 30:276–281, 2000.

[40] C. Vogler and D. Metaxas. Toward scalability in asl recognition: breaking down signs into phonemes. In *Proc. Intl. Gesture Workshop*, pages 400–404, 1999.

[41] T. Yamaguchi, M. Yoshihara, M. Akiba, M. Kuga, N. Kanazawa, and K. Kamata. Japanese sign language recognition system using information infrastructure. In *Proc. Joint 4th IEEE Intl. Conf. Fuzzy Systems, 2nd Intl. Fuzzy Eng. Symp.*, volume 5, pages 65–66, Yokohama, 1995.

[42] L. A. Zadeh. Theory of approximate reasoning. In J. Hayes, D. Michie, and L. I. Mikulich, editors, *Machine Intelligence*, pages 149–194. Ellis Horwood, 1970.

[43] L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Comm. ACM*, 37:77–84, 2006.

Analyzing Properties of Fuzzy Implications Obtained via the Interval Constructor

Benjamin C. Bedregal, Regivan H. N. Santiago
Federal University of Rio Grande do Norte
Department of Informatics and Applied Mathematics
Campus Universitário s/n, 59.072-970 Natal, Brazil,
{benjamin, regivan}@dimap.ufrn.br

Renata H. S. Reiser, Graçaliz P. Dimuro
Catholic University of Pelotas
Pos-Graduate Programme in Computer Science
Rua Feliz da Cunha, 412, 96.010-000 Pelotas, Brazil
{reiser,liz}@ucpel.tche.br

Abstract

This work considers an interval extension of fuzzy implications based on the best interval representation of usual fuzzy implications. The related properties of fuzzy implications can be naturally extended and the interval representation meets the optimality property and preserves the behaviors of the implications in the interval endpoints. Our discussion mainly focuses on the best interval representation of three important classes of fuzzy implications: S-implications, R-implications and QL-implications. We analyze sufficient and necessary conditions for these three classes of implications as inclusion-monotonic functions in both arguments satisfying the minimal properties of fuzzy implications.

1. Introduction

Interval-valued fuzzy sets were introduced independently by Zadeh [39] and others authors [18, 21, 31] in the seventies. This integration of Fuzzy Theory [38] with Interval Mathematics [26] has been studied from different viewpoints [12, 13, 33, 23, 28, 37, 29, 27, 17]. For example, Lodwick at [25] points out four ways to integrate fuzzy and interval approaches. One of them uses membership functions with intervals values, in order to model the uncertainty in the process of determining exact membership grades with the usual fuzzy membership functions.

Fuzzy implications play an important role in fuzzy logic, both in the broad sense (heavily applied to fuzzy control, analysis of vagueness in natural language and techniques of soft-computing) and in the narrow sense (developed as a branch of many-valued logic that is able to investigate deep logical questions). However, there is no consensus among researchers which extra properties fuzzy implications should satisfy. In the literature, several fuzzy implication properties have already been considered and their interrelationship with the other kinds of connectives are generally presented (see, e.g., [8]).

On the other hand, Santiago et al. [32] formalized two of some usual notions of interval computations, namely, the notions of correctness and optimality, explicitly required in [19]. That work described how those notions could be related to the notion of continuity on the real numbers and also to the notions of continuity proposed by Moore and Scott. The concept of *representation*, meaning "correct function", was introduced to emphasize the idea that interval computations represent computationally real entities. The most important result of that paper is the notion of canonical interval representation, which is an interval function that is optimal and correct. The resulting relations of those concepts with the usual continuity on the real numbers pointed out a method to extend real functions to suitable interval functions that preserve some desired properties, mainly, the continuity in terms of Moore and Scott.

Following the idea of extracting interval counterparts from real functions, using the approach above, Bedregal and Takahashi [6, 7] provided interval counterparts for fuzzy connectives. However, for fuzzy implications [7], the authors considered just those properties proposed by Fodor and Roubens [16] and the classes of R-implications. In this

paper, we also consider S-implications and QL-implications and prove that they are preserved by their canonical interval representations.

The paper is organized as follows. In Sect. 2, the definition of the best interval representation of a real function is summarized. Interval extensions of fuzzy t-norms, t-conorms and fuzzy negation are discussed in Sect. 3. Section 4 provides an analysis of properties of fuzzy implication. Section 5 shows that the minimal properties of fuzzy implications may be extended from interval fuzzy degrees. Interval fuzzy implications generated by interval fuzzy connectives are discussed in Sect. 6. Sect. 7 is the Conclusion.

2 Best Interval Representations

Consider the real unit interval $U = [0, 1] \subseteq \mathbb{R}$. Let $\mathbb{U} = \{[a, b] \mid 0 \leq a \leq b \leq 1\}$ be the set of subintervals of U. An interval has two projections $l : \mathbb{U} \longrightarrow U$ and $r : \mathbb{U} \longrightarrow U$, defined by $l([a, b]) = a$ and $r([a, b]) = b$, respectively. Denote $l(X)$ and $r(X)$ by $\underline{X}$ and $\overline{X}$, respectively.

Several natural partial orders may be defined on $\mathbb{U}$ [9]. The most used orders in the context of interval mathematics that are considered in this work are:

1. *Kulisch-Miranker or Product*: $X \leq Y$ if and only if $\underline{X} \leq \underline{Y}$ and $\overline{X} \leq \overline{Y}$.
2. *Inclusion order*: $X \subseteq Y$ if and only if $\underline{X} \geq \underline{Y}$ and $\overline{X} \leq \overline{Y}$

Definition 2.1 *Considering $X \in \mathbb{U}$ and $x \in U$, X represents x if $x \in X$. Given $X, Y \in \mathbb{U}$ and $x \in X \cap Y$, X is a* better representation *of x than Y, if $X \subseteq Y$* [1]. *A function $F : \mathbb{U}^n \longrightarrow \mathbb{U}$ is an* interval representation *of a function $f : U^n \longrightarrow U$ if, for each $\vec{X} \in \mathbb{U}^n$ and $\vec{x} \in \vec{X}$, $f(\vec{x}) \in F(\vec{X})$* [32]. *An interval function $F : \mathbb{U}^n \longrightarrow \mathbb{U}$ is a* better interval representation *of the function $f : U^n \longrightarrow U$ than $G : \mathbb{U}^n \longrightarrow \mathbb{U}$, denoted by $G \sqsubseteq F$, if, for each $\vec{X} \in \mathbb{U}^n$, the inclusion $F(\vec{X}) \subseteq G(\vec{X})$ holds.*

These simple notions emphasizes the idea that interval mathematics is a kind of language that expresses or describes real numbers and their associated functions.

Definition 2.2 *For each real function $f : U^n \longrightarrow U$, the interval function $\widehat{f} : \mathbb{U}^n \longrightarrow \mathbb{U}$, defined by*

$$\widehat{f}(\vec{X}) = [\inf\{f(\vec{x}) : \vec{x} \in \vec{X}\}, \sup\{f(\vec{x}) : \vec{x} \in \vec{X}\}], \quad (1)$$

is called the best interval representation *of f* [32].

The interval function $\widehat{f}$ is well defined and for any other interval representation F of f, $F \sqsubseteq \widehat{f}$. $\widehat{f}$ returns a narrower interval than any other interval representation of f, i.e., $\widehat{f}$ is the optimal representation of f (see Hickey et al. [19]).

Although the range of real functions applied to intervals, $f([a, b])$, can be seen as an operator (see [26], p.19) that preserves interesting properties of real functions, sometimes the resulting value is not an interval, and, thus, it is not a valid object in Moore arithmetic. Since we aim at to obtain an operator that transforms real functions into interval functions, the range is not a suitable operator for this purpose. The range $f([a, b])$ and the best interval representation $\widehat{f}[a, b]$ coincide only when f is continuous: if f is continuous, then for each $\vec{X} \in \mathbb{U}^n$, $\widehat{f}(\vec{X}) = \{f(\vec{x}) : \vec{x} \in \vec{X}\} = f(\vec{X})$.

An interval can be seen as a set of real numbers, or as a kind of number and also as an information of a real number. Each of these notions implies a way to classify intervals and to establish a criteria of proximity, namely, a topology. Seen as a kind of number, the associated topology is called Moore Topology, which is obviously an inheritance of a topology established on the Euclidean Plane, where the standard notion of proximity is defined in terms of the distance:

Definition 2.3 (Moore Topology) *Given two intervals $[a, b], [c, d] \in \mathbb{IR}$, the distance between $[a, b]$ and $[c, d]$ is defined by*

$$d([a, b], [c, d]) = max(\mid a - c \mid, \mid b - d \mid). \quad (2)$$

Seen as an information about a real number x, the criteria of proximity is established using quasi-metrics [1]. The resulting topology and the resulting notion of continuity is called Scott topology and Scott continuity, respectively:

Definition 2.4 (Scott Topology) *Given two intervals $[a, b], [c, d] \in \mathbb{IR}$, the quasi-distance between $[a, b]$ and $[c, d]$ is defined by*

$$q([a, b], [c, d]) = max(c - a, b - d, 0). \quad (3)$$

These notions of continuity and proximity for intervals depend on the chosen interpretation of an interval, and its respective influence on the algorithm convergence. The study of these relations and of the associated topology with respect to the viewpoint of "intervals as set of real numbers" is under analysis in the paper [5]. The relation between the continuity on real numbers and the above continuities is stated in the following theorem, proved in [32](p. 240).

Theorem 2.1 *Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a function. The following statements are equivalent:*

(i) *f is continuous;*

(ii) *$\widehat{f}$ is Scott continuous;*

(iii) *$\widehat{f}$ is Moore continuous.*

Clearly, Theorem 2.1 can be adapted to our context, i.e., considering U^n instead of $\mathbb{R}$.

[1] Trivially, this notion could be extended for tuples of intervals.

3 Interval t-norms, interval t-conorms and interval fuzzy negations

The generalizations proposed in [6] applies the principles discussed in the previous section. An interval t-norm (t-conorm) is considered an interval representation of a t-norm (t-conorm). This generalization fits with the idea that interval membership degrees may be thought as approximations of exact degrees [37].

Observe that a triangular norm (conorm), *t-norm* (*t-conorm*) for short, is a function $T : U^2 \to U$ that is commutative, associative, monotonic and has 1 (0) as identity.

A function $N : U \to U$ is a *fuzzy negation* if

N1: $N(0) = 1$ and $N(1) = 0$.

N2: If $x \geq y$ then $N(x) \leq N(y), \forall x, y \in I$.

In addition, fuzzy negations satisfying the involutive property are called *strong fuzzy negations* [22, 8]:

N3: $N(N(x)) = x, \forall x \in U$.

When the t-norm is considered, it is also possible to establish a partial order on fuzzy negations in a natural way. Let N_1 and N_2 be fuzzy negations. Then:

$$N_1 \leq N_2 \text{ if } \forall x \in U : N_1(x) \leq N_2(x). \quad (4)$$

Remark 3.1 *If $N_1 \leq N_2$ and $x \geq y$ then $N_1(x) \leq N_2(y)$.*

In the following, we show how to extend the concepts presented above by using the notion of best interval representation, according to [6, 7].

Definition 3.1 *A function $\mathbb{T} : \mathbb{U}^2 \to \mathbb{U}$ is an* interval t-norm *(*interval t-conorm*) if it is commutative, associative, monotonic with respect to the product and inclusion order, and has $[1, 1]$ ($[0, 0]$) as identity.*

Proposition 3.1 *If T is a t-norm (t-conorm) then $\widehat{T} : \mathbb{U}^2 \to \mathbb{U}$ is an interval t-norm.*

Definition 3.2 *An interval function $\mathbb{N} : \mathbb{U} \longrightarrow \mathbb{U}$ is an* interval fuzzy negation *if, for any X, Y in $\mathbb{U}$, the following properties hold:*

$\mathbb{N}$1: $\mathbb{N}([0, 0]) = [1, 1]$ *and* $\mathbb{N}([1, 1]) = [0, 0]$*;*

$\mathbb{N}$2: *If $X \geq Y$ then $\mathbb{N}(X) \leq \mathbb{N}(Y)$;*

$\mathbb{N}$3: *If $X \subseteq Y$ then $\mathbb{N}(X) \supseteq \mathbb{N}(Y)$.*

If $\mathbb{N}$ also satisfies the involutive property, then it is a strong interval fuzzy negation*:*

$\mathbb{N}$4: $\mathbb{N}(\mathbb{N}(X)) = X, \forall X \in \mathbb{U}$.

Theorem 3.1 *Let $N : U \longrightarrow U$ be a fuzzy negation. Then $\widehat{N}$ is an interval fuzzy negation. In addition, if N is a strong fuzzy negation then $\widehat{N}$ is a strong interval fuzzy negation.*

Proof: It is immediate that $\mathbb{N}1$ is satisfied. Also:

$\mathbb{N}$2: If $X \geq Y$ then $\overline{Y} \leq \overline{X}$ and $\underline{Y} \leq \underline{X}$. Therefore, by N2, it follows that:

$$\widehat{N}(X) = [N(\overline{X}), N(\underline{X})] \leq [N(\overline{Y}), N(\underline{Y})] = \widehat{N}(Y).$$

$\mathbb{N}$3: If $X \subseteq Y$ then $\overline{X} \leq \overline{Y}$ and $\underline{Y} \leq \underline{X}$. Therefore, by N2, it follows that:

$$\widehat{N}(X) = [N(\overline{X}), N(\underline{X})] \subseteq [N(\overline{Y}), N(\underline{Y})] = \widehat{N}(Y).$$

$\mathbb{N}$4: If $\mathbb{N}$ is strong, then

$$\begin{aligned} \widehat{N}(\widehat{N}(X)) &= \widehat{N}([N(\overline{X}), N(\underline{X})]) \\ &= [N(N(\underline{X})), N(N(\overline{X}))] = X. \ \diamond \end{aligned}$$

4 Fuzzy implication

Several definitions for fuzzy implication together with related properties have been studied (see, e.g., [2, 4, 8, 14, 16, 20, 24, 30, 34, 35, 36]). The unique consensus in these definitions is that the fuzzy implication should behave analogously as the classical implication does for the crisp case. Then, a binary function $I : U^2 \longrightarrow U$ is a *fuzzy implication* if I meets the minimal boundary conditions:

$$I(1, 1) = I(0, 1) = I(0, 0) = 1 \text{ and } I(1, 0) = 0. \quad (5)$$

Several reasonable extra properties that may be required for fuzzy implications are listed below:

I1: If $x \leq z$ then $I(x, y) \geq I(z, y)$;

I2: If $y \leq z$ then $I(x, y) \leq I(x, z)$;

I3: $I(0, y) = 1$ (falsity principle);

I4: $I(x, 1) = 1$ (right neutrality principle);

I5: $I(1, x) = x$ (left neutrality principle);

I6: $I(x, I(y, z)) = I(y, I(x, z))$ (exchange principle);

I7: $x \leq y$ if and only if $I(x, y) = 1$ (boundary condition);

I8: $I(x, x) = 1$ (identity property);

I9: $I(x, y) \geq y$;

I10: I is a continuous function (continuity property);

I11: $I(x, y) = I(x, I(x, y))$.

Other two properties related to fuzzy implications with strong negation may be also considered [8]:

I12: If N is a strong negation, then the contrapositive property holds: $I(x,y) = I(N(y), N(x))$.

I13: Consider $N : U \longrightarrow U$. If $N(x) = I(x,0)$ then N is a strong fuzzy negation.

In order to connect some fuzzy implications with some t-norms, an interesting study related to the law of importation is considered in [4]:

I14: Let T be a t-norm. Then, the law of importation concerned with T holds: I(T(x,y),z)=I(x,I(y,z)).

4.1 Generating fuzzy implications from fuzzy connectives

There are three usual ways to generate fuzzy implications from the other connectives. Let T be a t-norm, S be a t-conorm and N be a fuzzy negation. Then the equalities

$$I_T(x,y) = \sup\{z : T(x,z) \leq y\} \quad (6)$$

$$I_{S,N}(x,y) = S(N(x),y) \quad (7)$$

$$I_{T,S,N}(x,y) = S(N(x),T(x,y)) \quad (8)$$

are fuzzy implications, called R-implication or residuum of T, S-implication and QL-implication, respectively.

The R-implication arises from the notion of residuum in Intuitionistic Logic [3] or, equivalently, from the notion of residue in the theory of lattice-ordered semigroups [15]. This is well-defined only if the t-norm is left-continuous.

It is possible to define an S-implication from conjunction and negation (or disjunction and negation) using the corresponding tautology of classical logic. Thus, S-implications are based on the classical logical equivalence:

$$\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta. \quad (9)$$

Notice that, in some papers [8, 16, 15], an S-implication requires strong fuzzy negation. In this work this condition is not required, based on the approach considered in [22, 2].

QL-implications have the form used in quantum logic and are based on the"if-then-else" rules [15].

Several results about fuzzy implication and related to the properties **I1**, ..., **I14** may be studied. According to the results presented in [16], it is immediate that:

Proposition 4.1 *If $I : U^2 \rightarrow U$ is an R-implication then the function I satisfies the properties* **I2**, **I6**, **I7** *and* **I9**.

Based on the results presented in [3], it follows that:

Proposition 4.2 *When $I : U^2 \rightarrow U$ is an S-implication, the properties* **I1**, **I2**, **I5**, **I7** *and* **I9** *hold.*

Proposition 4.3 *If $I : U^2 \rightarrow U$ is an QL-implication then I satisfies the properties* **I2**, **I5** *and* **I12**.

5 Interval fuzzy implications

In Interval Mathematics, any value may be identified with a degenerate interval. Then, the minimal properties of fuzzy implications can be naturally extended for interval fuzzy degrees. A function $\mathbb{I} : \mathbb{U}^2 \longrightarrow \mathbb{U}$ is a *interval fuzzy implication* if the following conditions are satisfied:

$$\mathbb{I}([1,1],[1,1]) = \mathbb{I}([0,0],[0,0]) = \mathbb{I}([0,0],[1,1]) = [1,1],$$
$$\mathbb{I}([1,1],[0,0]) = [0,0].$$

The properties stated in the previous section can be naturally extended for intervals. Notice that, since we have two natural partial orders on $\mathbb{U}$ and two continuity notions, some properties can have two extensions.

5.1 Extended properties

$\mathbb{I}$1: If $X \leq Z$ then $\mathbb{I}(X,Y) \geq \mathbb{I}(Z,Y)$;

$\mathbb{I}$2: If $Y \leq Z$ then $\mathbb{I}(X,Y) \leq \mathbb{I}(X,Z)$;

$\mathbb{I}$3: $\mathbb{I}([0,0],Y) = [1,1]$;

$\mathbb{I}$4: $\mathbb{I}(X,[1,1]) = [1,1]$;

$\mathbb{I}$5: $\mathbb{I}([1,1],X) = X$;

$\mathbb{I}$6: $\mathbb{I}(X,\mathbb{I}(Y,Z)) = \mathbb{I}(Y,\mathbb{I}(X,Z))$;

$\mathbb{I}$7: $\overline{X} \leq \underline{Y}$ if and only if $\mathbb{I}(X,Y) = [1,1]$;

$\mathbb{I}$8: $1 \in \mathbb{I}(X,X)$;

$\mathbb{I}$9: $\mathbb{I}(X,Y) \geq Y$;

$\mathbb{I}$10a: $\mathbb{I}$ is a Moore continuous function;

$\mathbb{I}$10b: $\mathbb{I}$ is a Scott continuous function;

$\mathbb{I}$11a: $\mathbb{I}(X,Y) \subseteq \mathbb{I}(X,\mathbb{I}(X,Y))$;

$\mathbb{I}$11b: $\mathbb{I}([x,x],Y) = \mathbb{I}([x,x],\mathbb{I}([x,x],Y))$;

$\mathbb{I}$12: Let $\mathbb{N}$ be a strong fuzzy negation. When $\mathbb{I}$ is contrapositive with respect to $\mathbb{N}$, then $\mathbb{I}(X,Y) = \mathbb{I}(\mathbb{N}(Y),\mathbb{N}(X))$.

$\mathbb{I}$13: If $\mathbb{N} : \mathbb{U} \longrightarrow \mathbb{U}$, $\mathbb{N}(X) = \mathbb{I}(X,[0,0])$ then $\mathbb{N}$ is a strong interval fuzzy negation.

$\mathbb{I}$14: $\mathbb{I}(\mathbb{T}(X,Y),Z) = \mathbb{I}(X,\mathbb{I}(Y,Z))$, if $\mathbb{T}$ is an interval t-norm.

From any fuzzy implication it is always possible to obtain an interval fuzzy implication canonically. The interval fuzzy implication, obtained in this way, also meets the optimality property and preserves the same properties satisfied by the fuzzy implication. In the following two propositions, the best interval representation of fuzzy implication

is shown as an inclusion-monotonic function in both arguments and the related proofs can be constructed straightforward from the definition of $\widehat{I}$.

Proposition 5.1 *If I is a fuzzy implication then $\widehat{I}$ is an interval fuzzy implication.*

Proposition 5.2 *Let I be a fuzzy implication. Then for each $X_1, X_2, Y_1, Y_2 \in \mathbb{U}$. If $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$ then $\widehat{I}(X_1, Y_1) \subseteq \widehat{I}(X_2, Y_2)$.*

Theorem 5.1 *Let I be a fuzzy implication. If I satisfies a property* **Ik***, for some* k$= 1, \ldots, 10$*, then $\widehat{I}$ satisfies the property* $\mathbb{I}$**k**.

Proof:

$\mathbb{I}$**1:** If $u \in \widehat{I}(X, Y)$, then there exist $x \in X$ and $y \in Y$ such that $I(x, y) = u$. If $X \leq Z$, then there exists $z \in Z$ and $x \leq z$. So, by **I1**, $u = I(x, y) \geq I(z, y)$. On the other hand, if $v \in \widehat{I}(Z, Y)$, then there exist $z \in Z$ and $y \in Y$ such that $I(z, y) = v$. If $X \leq Z$, then $x \leq z$, for some $x \in X$. So, by **I1**, $I(x, y) \geq I(z, y) = v$. Therefore, for each $u \in \widehat{I}(X, Y)$, there is $v \in \widehat{I}(Z, Y)$ and $u \geq v$. In addition, for each $v \in \widehat{I}(Z, Y)$, there is $u \in \widehat{I}(X, Y)$ such that $u \geq v$. Hence, $\widehat{I}(X, Y) \geq \widehat{I}(Z, Y)$.

$\mathbb{I}$**2:** If $u \in \widehat{I}(X, Y)$, then there exist $x \in X$ and $y \in Y$ such that $I(x, y) = u$. If $Y \leq Z$, then there exists $z \in Z$ such that $y \leq z$. So, by **I2**, $u = I(x, y) \leq I(x, z)$. On the other hand, if $v \in \widehat{I}(X, Z)$, then there exist $z \in Z$ and $x \in X$ such that $I(x, z) = v$. If $Y \leq Z$, then $y \leq z$, for some $y \in Y$. So, by **I2**, $I(x, y) \geq I(x, z) = v$. Therefore, for each $u \in \widehat{I}(X, Y)$, there is $v \in \widehat{I}(X, Z)$ such that $u \leq v$, and, for each $v \in \widehat{I}(X, Z)$,there is $u \in \widehat{I}(X, Y)$ such that $u \leq v$. Hence, $\widehat{I}(X, Y) \leq \widehat{I}(X, Z)$.

$\mathbb{I}$**3:** Trivially, by **I3**, for each $y \in Y$, $I(0, y) = 1$, and then $\{I(0, y) : y \in Y\} = [1, 1]$. Thus, since $\widehat{I}([0, 0], Y)$ is the narrowest interval containing $\{I(0, y) : y \in Y\}$, then $\widehat{I}([0, 0], Y) = [1, 1]$.

$\mathbb{I}$**4:** Trivially, by **I4**, for each $x \in X$, $I(x, 1) = 1$ and then $\{I(x, 1) : x \in X\} = [1, 1]$. Thus, since $\widehat{I}(X, [1, 1])$ is the narrowest interval containing $\{I(x, 1) : x \in X\}$, then $\widehat{I}(X, [1, 1]) = [1, 1]$.

$\mathbb{I}$**5:** Trivially, by **I5**, for each $x \in X$, $I(1, x) = x$ and then $\{I(1, x) : x \in X\} = X$. Thus, since $\widehat{I}([1, 1], X)$ is the narrowest interval containing $\{I(1, x) : x \in X\}$, then $\widehat{I}([1, 1], X) = X$.

$\mathbb{I}$**6:** If $u \in \widehat{I}(X, \widehat{I}(Y, Z))$ then there exist $x \in X$, $y \in Y$ and $z \in Z$ such as $I(x, I(y, z)) = u$. But, by **I6**, $u = I(y, I(x, z))$. So, $u \in \widehat{I}(Y, \widehat{I}(X, Z))$ and, therefore, $\widehat{I}(X, \widehat{I}(Y, Z)) \subseteq \widehat{I}(Y, \widehat{I}(X, Z))$. Analogously, if $u \in \widehat{I}(Y, \widehat{I}(X, Z))$ then there exist $x \in X$, $y \in Y$ and $z \in Z$ such that $I(y, I(yx, z)) = u$. But, by **I6**, $u = I(x, I(y, z))$. So, $u \in \widehat{I}(X, \widehat{I}(XY, Z))$ and, therefore, $\widehat{I}(Y, \widehat{I}(X, Z)) \subseteq \widehat{I}(X, \widehat{I}(Y, Z))$. Hence, $\widehat{I}(X, \widehat{I}(Y, Z)) = \widehat{I}(Y, \widehat{I}(X, Z))$.

$\mathbb{I}$**7:** If $\overline{X} \leq \underline{Y}$ then for each $x \in X$ and $y \in \underline{Y}$, $x \leq y$, and then, by **I7**, $I(x, y) = 1$. Therefore, $\widehat{I}(X, Y) = \{I(x, y) : x \in X \text{ and } y \in Y\} = \{1\} = [1, 1]$. Conversely, if $\widehat{I}(X, Y) = [1, 1]$, then $\{I(x, y) : x \in X \text{ and } y \in Y\} = \{1\}$, and then, for each $x \in X$ and $y \in Y$, $I(x, y) = 1$. Thus, $I(\overline{X}, \underline{Y}) = 1$, and hence, by **I7**, $\overline{X} \leq \underline{Y}$.

$\mathbb{I}$**8:** If $x \in X$ then $I(x, x) = 1$, and then $1 \in \widehat{I}(X, X)$.

$\mathbb{I}$**9:** By **I9**, for each $x \in X$ and $y \in Y$, $I(x, y) \geq y$. So, $\widehat{I}(X, Y) \geq Y$.

$\mathbb{I}$**10a,**$\mathbb{I}$**10b:** It is straightforward, from Theorem 2.1.

$\mathbb{I}$**11a:** If $u \in \widehat{I}(X, Y)$, then there exist $x \in X$ and $y \in Y$ such that $I(x, y) = u$. So, by **I11**, $u = I(x, I(x, y))$, and, therefore, $u \in \widehat{I}(X, \widehat{I}(X, Y))$. Hence, $\widehat{I}(X, Y) \subseteq \widehat{I}(X, \widehat{I}(X, Y))$.

$\mathbb{I}$**11b:** By $\mathbb{I}$**10a**, $\widehat{I}([x, x], Y) \subseteq \widehat{I}([x, x], \widehat{I}([x, x], Y))$. So, it remains to prove that $\widehat{I}([x, x], Y) \supseteq \widehat{I}([x, x], \widehat{I}([x, x], Y))$. Let $u \in \widehat{I}([x, x], \widehat{I}([x, x], Y))$, then there exists $y \in Y$ such that $u = I(x, I(x, y))$. But, by **I11**, $I(x, I(x, y)) = I(x, y)$. So, $u \in \widehat{I}([x, x], Y)$, and, therefore, $\widehat{I}([x, x], Y) \supseteq \widehat{I}([x, x], \widehat{I}([x, x], Y))$. ◇

The preservation of properties $\mathbb{I}$**12** – $\mathbb{I}$**14** will be proved separately, since another connective will be considered.

Proposition 5.3 *Let I be a fuzzy implication and N be a fuzzy strong negation, where I is contrapositive concerned with N, i.e., I satisfies* **I12***. Then $\widehat{I}$ is contrapositive concern with $\widehat{N}$, i.e., $\widehat{I}$ satisfies* $\mathbb{I}$**12**.

Proof: If $u \in \widehat{I}(X, Y)$, then there exist $x \in X$ and $y \in Y$ such that $I(x, y) = u$. But, by **I12**, $I(x, y) = I(N(y), N(x))$. Since $N(y) \in \widehat{N}(Y)$ and $N(x) \in \widehat{N}(X)$, then $u \in \widehat{I}(\widehat{N}(Y), \widehat{N}(X))$. So, $\widehat{I}(X, Y) \subseteq \widehat{I}(\widehat{N}(Y), \widehat{N}(X))$. On the other hand, if $u \in \widehat{I}(\widehat{N}(Y), \widehat{N}(X))$, then there exist $v \in \widehat{N}(Y)$ and $w \in \widehat{N}(X)$ such that $I(v, w) = u$. But, since $v \in \widehat{N}(Y)$

and $w \in \widehat{N}(X)$, there exist $y \in Y$ and $x \in X$, such that $N(y) = v$ and $N(x) = w$. So, $I(N(y), N(x)) = u$. But, by **I12**, $I(N(y), N(x)) = I(x, y)$. Then, $u \in \widehat{I}(X, Y)$ and $\widehat{I}(X, Y) = \widehat{I}(\widehat{N}(Y), \widehat{N}(X))$. ◇

Proposition 5.4 *Let I be a fuzzy implication. If I satisfies the property* **I13***, then the interval function $\mathbb{N} : \mathbb{U} \longrightarrow \mathbb{U}$, defined by $\mathbb{N}(X) = \widehat{I}(X, [0, 0])$, is a strong interval fuzzy negation, i.e., $\widehat{I}$ satisfies the property* $\mathbb{I}$**13**.

Proof: By **I13**, $N : U \longrightarrow U$, defined by $N(x) = I(x, 0)$, is a strong fuzzy implication, and, therefore, by Theorem 3.2, $\widehat{N}$ is a strong interval fuzzy negation. We will prove that $\mathbb{N} = \widehat{N}$. Consider $X \in \mathbb{U}$. If $u \in \mathbb{N}(X)$, then there exists $x \in X$ such that $I(x, 0) = u$, and, therefore $N(x) = u$. So, $u \in \widehat{N}(X)$. Conversely, if $u \in \widehat{N}(X)$, then there exists $x \in X$ and $N(x) = u$. But, by **I13**, $I(x, 0) = u$. So, $u \in \widehat{I}(X, [0, 0])$, i.e., $u \in \mathbb{N}(X)$. Therefore, $\mathbb{N} = \widehat{N}$. ◇

Proposition 5.5 *Let I be a fuzzy implication and T be a t-norm, with I satisfying the law of importation concerned with T (*$\mathbb{I}$**14***). Then $\widehat{I}$ satisfies the property* $\mathbb{I}$**14** *concerned with $\widehat{T}$.*

Proof: If $u \in \widehat{I}(\widehat{(T)}(X, Y), Z)$ then there exist $v \in \widehat{(T)}(X, Y)$ and $z \in Z$ with $u = I(v, z)$. But, if $v \in \widehat{(T)}(X, Y)$, then there exist $x \in X$ and $y \in Y$ such that $v = T(x, y)$. So, $u = I(T(x, y), z)$, and, therefore, by property **I14**, $u = I(x, I(y, z))$. Thus, since $x \in X$ and $I(y, z) \in \widehat{I}(Y, Z)$, $u \in \widehat{I}(X, \widehat{I}(Y, Z))$. Therefore, $\widehat{I}(\widehat{(T)}(X, Y), Z) \subseteq \widehat{I}(X, \widehat{I}(Y, Z))$. On the other hand, if $u \in \widehat{I}(X, \widehat{I}(Y, Z))$ then there exist $x \in X$ and $v \in \widehat{I}(Y, Z)$ such that $u = I(x, v)$. But, if $v \in \widehat{I}(Y, Z)$, then there exist $y \in Y$ and $z \in Z$ such that $v = I(y, z)$. So, $u = I(x, I(y, z))$, and, therefore, by property **I14**, $u = I(T(x, y), z)$. Thus, since $T(x, y) \in \widehat{T}(X, Y)$ and $z \in Z$, $u \in \widehat{I}(\widehat{(T)}(X, Y), Z)$. Therefore, $\widehat{I}(\widehat{(T)}(X, Y), Z) = \widehat{I}(X, \widehat{I}(Y, Z))$. ◇

6 Generating interval fuzzy implications from interval fuzzy connectives

In [10, 11, 17] it is possible to find some definitions of interval valued implications. However, the approach proposed here is in a different context. In this section, the interval fuzzy implications are generated from interval fuzzy connectives, obtained through the interval constructor.

6.1 Interval R-implications

An interval fuzzy implication $\mathbb{I}$ is an *interval R-implication* if there is an interval t-norm $\mathbb{T}$ defined as

$$\mathbb{I}(X, Y) = \sup\{Z \in \mathbb{U} : \mathbb{T}(X, Z) \leq Y\}. \tag{10}$$

In this case we denote it by $\mathbb{I}_{\mathbb{T}}$ instead of $\mathbb{I}$.

Proposition 6.1 *Let T be a t-norm. Then:*

$$\mathbb{I}_{\widehat{T}} = \widehat{I_T}. \tag{11}$$

Proof: See [7]. ◇

Proposition 6.2 *Let $\mathbb{I}$ be an interval fuzzy implication. If $\mathbb{I}$ is an interval R-implication then $\mathbb{I}$ satisfies* $\mathbb{I}$**2***,* $\mathbb{I}$**6***,* $\mathbb{I}$**7** *and* $\mathbb{I}$**9**.

Proof: Considering Def. 2.2 and Prop. 6.2, the proof can be constructed analogously to Prop. 4.1. ◇

6.2 Interval S-Implications

An interval fuzzy implication $\mathbb{I}$ is an *interval S-implication* ($\mathbb{I}_{\mathbb{S},\mathbb{N}}$) if there are an interval t-conorm $\mathbb{S}$ and an interval fuzzy negation $\mathbb{N}$ such that

$$\mathbb{I}(X, Y) = \mathbb{S}(\mathbb{N}(X), Y). \tag{12}$$

Proposition 6.3 *Let S be a t-conorm and N be a fuzzy negation. Then:*

$$\mathbb{I}_{\widehat{S},\widehat{N}} = \widehat{I_{S,N}}. \tag{13}$$

Proof: Considering $X, Y \in \mathbb{U}$, then

$$\begin{aligned}
\mathbb{I}_{\widehat{S},\widehat{N}}(X, Y) &= \widehat{S}(\widehat{N}(X), Y) \\
&= \widehat{S}([N(\overline{X}), N(\underline{X})], Y) \\
&= [S(N(\overline{X}), \underline{Y}), S(N(\underline{X}), \overline{Y})] \\
&= [I_{S,N}(\overline{X}, \underline{Y}), I_{S,N}(\underline{X}, \overline{Y})].
\end{aligned}$$

Therefore, $\mathbb{I}_{\widehat{S},\widehat{N}}(X, Y) = \widehat{I_{S,N}}(X, Y)$. ◇

Proposition 6.4 *Let $\mathbb{I}$ be an interval fuzzy implication. If $\mathbb{I}$ is an interval S-implication then $\mathbb{I}$ satisfies* $\mathbb{I}$**1***,* $\mathbb{I}$**2***,* $\mathbb{I}$**5***,* $\mathbb{I}$**7** *and* $\mathbb{I}$**9**.

Proof: It is analogous to Prop. 4.2. ◇

6.3 Interval QL-implications

An interval fuzzy implication $\mathbb{I}$ is an *interval QL-implication* ($\mathbb{I}_{\mathbb{T},\mathbb{S},\mathbb{N}}$) if there are an interval t-norm $\mathbb{T}$, an interval t-conorm $\mathbb{S}$ and an interval fuzzy negation $\mathbb{N}$ such that

$$\mathbb{I}(X,Y) = \mathbb{S}(\mathbb{N}(X), \mathbb{T}(X,Y)). \tag{14}$$

Proposition 6.5 *Let T be a t-norm, S be a t-conorm and N be a fuzzy negation. Then:*

$$\mathbb{I}_{\widehat{T},\widehat{S},\widehat{N}} = \widehat{I_{T,S,N}}. \tag{15}$$

Proof: We prove that $\widehat{I_{T,S,N}} \subseteq \mathbb{I}_{\widehat{T},\widehat{S},\widehat{N}}$. Considering $X, Y \in \mathbb{U}$, then

$$\begin{aligned}
&\mathbb{I}_{\widehat{S},\widehat{N},\widehat{T}}(X,Y) \\
&\quad = \widehat{S}(\widehat{N}(X), \widehat{T}(X,Y)) \\
&\quad = \widehat{S}([N(\overline{X}), N(\underline{X})], [T(\underline{X},\underline{Y}), T(\overline{X},\overline{Y})) \\
&\quad = [S(N(\overline{X}), T(\underline{X},\underline{Y})), S(N(\underline{X}), T(\overline{X},\overline{Y}))].
\end{aligned}$$

The results follows immediately, since, for each $x \in X, y \in Y$, it holds that

$$S(N(\overline{X}), T(\underline{X},\underline{Y})) \leq S(N(x), T(x,y)) \leq S(N(\underline{X}), T(\overline{X},\overline{Y})).$$

Analogously, it is possible to prove that $\mathbb{I}_{\widehat{T},\widehat{S},\widehat{N}} \subseteq \widehat{I_{T,S,N}}$.$\diamond$

Proposition 6.6 *Let $\mathbb{I}$ be an interval fuzzy implication. If $\mathbb{I}$ is an interval QL-implication then $\mathbb{I}$ satisfies $\mathbb{I}$**2**, $\mathbb{I}$**5** and $\mathbb{I}$**12**.*

Proof: It is analogous to Prop. 4.3. $\diamond$

7 Conclusion and Final Remarks

Following the ideas of [12, 33, 23, 28, 37, 29, 27, 25, 17], throughout this paper, intervals were used to model the uncertainty of a specialist's information related to truth values in the fuzzy propositional calculus. The basic systems are based on interval t-norm, i.e., using subsets of the real unit interval as the standard sets of truth degrees, continuous t-norms as standard truth interval functions of conjunction and their residua as standard truth interval functions of implication.

This paper summarizes the results presented in [6, 7]. As in the previous works, it applies the concept of best interval representation as a method for the construction of fuzzy interval connectives. It also introduces the classes of R-implications, S-implication and QL-implications. One of the main results in the paper is Theorem 5.1, which states that if a fuzzy implication satisfies a property, then its interval counterpart, built as its best interval representation, also satisfies the analogous property.

In addition, we discussed under which conditions generalized fuzzy implications applied to interval values preserve properties of canonical forms generated by interval t-norms (t-conorm). We strong urge the reader to become familiar with properties of R-implications, S-implications and QL-implications that are preserved by the interval representation.

This paper is not only useful to analyze deductive systems in mathematical depth, but it is also a foundation for methods of fuzzy logic in a broad sense, that means, a branch of many-valued logics based on the paradigm of inference under vagueness and imprecision.

8 Acknowledgments

This paper was partially supported by the Brazilian funding agencies CNPq (Proc. 470871/2004-0) and FAPERGS. We are very grateful to the referees for their valuable comments that helped us to improve the paper.

Referências

[1] B. Acióly and B. Bedregal. A quasi-metric topology compatible with inclusion monotonicity on interval space. *Reliable Computing*, 3(3):305–313, 1997.

[2] M. Baczynski. Residual implications revisited. Notes on the Smets-Magrez. *Fuzzy Sets and Systems*, 145(2):267–277, 2004.

[3] M. Baczynski and B. Jayaran. On the characterization of (S,N)-implications generated from continuous negations. In *Proc. 11th Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 436–443, Paris, 2006.

[4] J. Balasubramaniam. Yager's new class of implications J_f and some classical tautologies. *Information Sciences*, 2006.

[5] B. Bedregal and R. Santiago. Characterizing and specifying optimal and correct interval functions. *Formal Aspects of Computing*, 2007. (Submitted).

[6] B. Bedregal and A. Takahashi. The best interval representation of t-norms and automorphisms. *Fuzzy Sets and Systems*, 157(24):3220–3230, 2006.

[7] B. Bedregal and A. Takahashi. Interval valued versions of t-conorms, fuzzy negations and fuzzy implications. In *IEEE Proc. Intl. Conf. Fuzzy Systems*, Vancouver-Canada, 2006.

[8] H. Bustince, P. Burilo, and F. Soria. Automorphism, negations and implication operators. *Fuzzy Sets and Systems*, 134:209–229, 2003.

[9] R. Callejas-Bedregal and B. Bedregal. Intervals as a domain constructor. *TEMA*, 2(1):43 – 52, 2001. (Available at http://www.sbmac.org.br/tema).

[10] G. Cornelis, G. Deschrijver, and E. Kerre. Implication in intuitionistic fuzzy and interval-valued fuzzy set theory: construction, classification, application. *Intl. Journal Approximate Reason.*, 35:55–95, 2004.

[11] G. Deschrijver and E. Kerre. Implicators based on binary aggregation operators in interval-valued fuzzy set theory. *Fuzzy Sets and Systems*, 153(2):229–248, 2005.

[12] D. Dubois and H. Prade. *Fuzzy Sets and Systems*. Academic Press, New York, 1996.

[13] D. Dubois and H. Prade. Interval-valued fuzzy sets, possibility theory and imprecise probability. In *Proc. Intl. Conf. Fuzzy Logic and Technology*, pages 314–319, Barcelona, 2005.

[14] J. Fodor. On fuzzy implication operators. *Fuzzy Sets and Systems*, 42:293–300, 1991.

[15] J. Fodor. Contrapositive symmetry of fuzzy implications. *Fuzzy Sets and Systems*, 69:141–156, 1995.

[16] J. Fodor and M. Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publisher, Dordrecht, 1994.

[17] B. V. Gasse, G. Cornelis, G. Deschrijver, and E. Kerre. On the properties of a generalized class of t-norms in interval-valued fuzzy logics. *New Mathematics and Natural Computation*, 2:29–42, 2006.

[18] I. Grattan-Guiness. Fuzzy membership mapped onto interval and many-valued quantities. *Z. Math. Logik. Grundladen Math.*, 22:149–160, 1975.

[19] T. Hickey, Q. Ju, and M. Emdem. Interval arithmetic: from principles to implementation. *Journal of the ACM*, 48(5):1038–1068, 2001.

[20] R. Horcik and M. Navara. Validation sets in fuzzy logics. *Kybernetika*, 38(2):319–326, 2002.

[21] K. Jahn. Intervall-wertige mengen. *Math. Nach.*, 68:115–132, 1975.

[22] E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publisher, Dordrecht, 2000.

[23] G. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logics: Theory and Applications*. Prentice Hall, Upper Saddle River, 1995.

[24] J. Leski. ε-insensitive learning techniques for approximate reasoning system. *Int. Jour. of Computational Cognition*, 1(1):21–77, 2003.

[25] W. Lodwick. Preface. *Reliable Computing*, 10(4):247–248, 2004.

[26] R. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[27] R. Moore and W. Lodwick. Interval analysis and fuzzy set theory. *Fuzzy Sets and Systems*, 135(1):5–9, 2003.

[28] H. Nguyen, V. Kreinovich, and Q. Zuo. Interval-valued degrees of belief: applications of interval computations to expert systems and intelligent control. *Intl. Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems*, 5(3):317–358, 1997.

[29] H. Nguyen and E. Walker. *A First Course in Fuzzy Logic*. Chapman & Hall/CRC, Boca Raton, 1999.

[30] D. Ruan and E. Kerre. Fuzzy implication operators and generalized fuzzy methods of cases. *Fuzzy Sets and Systems*, 54:23–37, 1993.

[31] R. Sambuc. *Fonctions ϕ-floues. Application l'aide au diagnostic en pathologie thyroidienne*. PhD thesis, Univ. Marseille, Marseille, 1975.

[32] R. Santiago, B. Bedregal, and B. Acióly. Formal aspects of correctness and optimality in interval computations. *Formal Aspects of Computing*, 18(2):231–243, 2006.

[33] I. Turksen. Interval valued fuzzy sets based on normal forms. *Fuzzy Sets and Systems*, 20:191–210, 1986.

[34] R. Yager. On the implication operator in fuzzy logic. *Information Sciences*, 31:141–164, 1983.

[35] R. Yager. On global requirements for implication operators in fuzzy modus ponens. *Fuzzy Sets and Systems*, 106:3–10, 1999.

[36] R. Yager. On some new classes of implication operators and their role in approximate reasoning. *Information Sciences*, 167:193–216, 2004.

[37] Y. Yam, M. Mukaidono, and V. Kreinovich. Beyond [0,1] to intervals and further: Do we need all new fuzzy values? In *Proc. 8th Intl. Fuzzy Systems Assoc. World Congress*, pages 143–146, Taipei, 1999.

[38] L. A. Zadeh. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

[39] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning - I. *Information Sciences*, 6:199–249, 1975.

Interval Analysis of Linear Analog Circuits

Alexander Dreyer
Fraunhofer Institute for Industrial Mathematics (ITWM)
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
alexander.dreyer@itwm.fraunhofer.de

Abstract

Reliable methods for the analysis of tolerance-affected analog circuits are of great importance in nowadays microelectronics. It is impossible to produce circuits with exactly those parameter specifications proposed in the design process. Interval arithmetic can be used to obtain a worst-case analysis of the influence of component tolerances.

This paper focuses on a new approach for interval-valued frequency-response analysis of linear analog circuits, which consist of current and voltage sources as well as resistors, capacitances, inductances, and all variants of controlled sources. Part and parcel of this strategy is the handling of fill-in patterns for those parameters related to uncertain components. Such systems can efficiently be solved by successive application of the Sherman-Morrison formula. The approach is also extended to complex-valued systems from frequency-domain analysis of linear circuits. Crude bounds can be obtained by treating real and imaginary part as different variables. The latter is improved by considering the correlations in order to obtain tighter enclosures of the solution.

1 Motivation

Numerical simulations of analog circuits can be used to analyze a circuit's behaviour without the need for a physical implementation. But actual circuit properties may differ from the results obtained by floating-point simulations, due to errors caused by rounding, component tolerances, and simplified models. Simulations based on interval arithmetic can be used as a unified framework to bound all these errors, but tend to be too conservative.

In this paper a new approach for computing tight bounds for a worst-case analysis of tolerance-affected analog circuits is introduced. In this area, component tolerances result from the fact, that the desired properties cannot be met exactly during manufacturing of the actual circuit. For this purpose mathematical methods will be described, which are tuned especially to handle systems arising from linear analog circuits. This includes an approach, which is capable of computing tight bounds to a frequency-response analysis with respect to variations due to a large number of tolerance-affected components. Finally, the results are illustrated by giving an example application.

2 Worst-case Analysis and Intervals

Conventional methods for the worst-case analysis of tolerance-affected parameters include sampling and the Monte Carlo method. In the sampling method, the range of the behavior of the circuit is approximated by solving the circuit equations for a finite number of points in each parameter interval.

The Monte Carlo approach is a statistical method, which has the ability to simulate the real production process. The generation of random numbers with respect to given probability distribution of the parameters can be used to generate a sufficiently large number of parameter sets. This results in a large number of simulations, which can be used to measure the circuit performance spread and to characterize the response statistically [11]. Using a uniform distribution of the component tolerances, worst-case results can be estimated with this method.

Alternatively, upper and lower bounds for the uncertain parameters can be interpreted as the endpoints $\underline{x}$, $\overline{x}$ of a closed interval $[\underline{x}, \overline{x}] \subseteq \mathbb{R}$, such that *interval arithmetic* is applicable (e. g. see [6, 8]). During computation of a desired result any expression is constructed by subsequent calls of binary operations and basic functions. For each of which, outer bounds to the range can be computed by exploiting the respective monotonic or piecewise monotonic properties. Conservative approximations for more complex expressions can be evaluated by combination of these elementary functions.

Following, an interval-valued magnitude is denoted by $[x]$; a vector of intervals – or *box* – is consequently written as $[\mathbf{x}]$, while $[\mathbb{R}]$ is the set of intervals over the reals.

3 Fill-in Patterns of Linear Circuits

In the case of linear analog circuits with uncertain parameters Kirchhoff's laws and element relations are summarized in a matrix equation of the following form:

$$\mathbf{A}(\mathbf{p}) \cdot \mathbf{x} = \mathbf{b}\,, \tag{1}$$

where $\mathbf{x}$ denotes the vector of internal currents and voltages, and $\mathbf{p} = (p_1, \ldots, p_{n_p})$ corresponds to tolerance-affected components, which are bounded by intervals $[p_i]$. Solving such an interval equation system means determining close bounds to the smallest box $[\mathbf{x}^*]$ with

$$[\mathbf{x}^*] \supseteq \left\{ (\mathbf{A}(\mathbf{p}))^{-1} \cdot \mathbf{b} \,\middle|\, p_i \in [p_i] \right\}. \tag{2}$$

Note that uncertain values of independent current and voltage sources can also be modeled as interval-valued parameters on the right-hand side $\mathbf{b}$. These kind of parameters can be moved to the matrix by the cost of introducing new rows and columns.

Earlier efforts for solving interval-valued linear circuit equations, were already capable of computing outer bounds to $[\mathbf{x}^*]$, for instance see [9], but also [10] and [14]. But these did not utilize the special matrix structure arising from the analysis of analog circuits. Therefore, a new approach was developed, which obeys this additional information. First of all we will assume that the parameter dependence of $\mathbf{A}(\mathbf{p})$ can be written as a sequence of rank-one updates of a parameter-independent matrix $\mathbf{A}_0 \in \mathbb{R}^{n \times n}$:

$$\mathbf{A}(\mathbf{p}) = \mathbf{A}_0 + \sum_{i=1}^{n_\mathbf{p}} p_i \cdot (\mathbf{u}_i \cdot \mathbf{v}_i^\mathsf{T})\,, \tag{3}$$

with $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^n$, and $\mathbf{A}(\mathbf{p})$ is invertible for all $\mathbf{p} \in [\mathbf{p}]$. This is not a restriction at all, because this structure is already inherent to a linear circuit. Using the *sparse tableau formulation* (*STA*) [15] to generate the linear circuit equations, each p_i will occur only once and $\mathbf{u}_i, \mathbf{v}_i$ are just unit vectors, which define the corresponding matrix element. In the case of *modified nodal analysis* (*MNA*) [15] the matrices $p_i \cdot (\mathbf{u}_i \cdot \mathbf{v}_i^\mathsf{T})$ correspond to the well-known fill-in patterns used during equation setup.

4 Mathematical Methods

In this section, a method for treating uncertain linear circuit elements is proposed. As seen in Section 3, for most common formulations for circuit equations result in linear systems in fill-in pattern form of Equation 2. It has already been pointed out independently by Dreyer [1] and Ganesan et al. [5], that this form emits useful monotonicity property. Furthermore, it can be utilized for efficient solving of interval-valued circuit equations [2], see also [3] for more details and proofs.

4.1 Methods for Resistive Circuits

The *Sherman-Morrison formula* can be used to calculate the inverse of a perturbed matrix. Under certain conditions repeated inversions can be avoided by reusing the original inverse.

Theorem 4.1 (Sherman-Morrison) *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be invertible and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. Then the matrix $\mathbf{A} + \mathbf{u} \cdot \mathbf{v}^\mathsf{T}$ is invertible if and only if $1 + \mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u} \neq 0$. In this case we have:*

$$(\mathbf{A} + \mathbf{u} \cdot \mathbf{v}^\mathsf{T})^{-1} = \mathbf{A}^{-1} - \frac{1}{1 + \mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u}} \mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\,. \tag{4}$$

Equation 4 has already been used in the field of analog circuits analysis for calculating the influence of a single matrix entry to the solution [7, 16]. It can immediately be used to analyze the pertubations of $\mathbf{A}(p)^{-1}\mathbf{b}$ with respect to a single uncertain parameter p, which may vary in the interval $[\underline{p}, \overline{p}]$. The corresponding fill-in pattern can then be written as $\mathbf{A}(p) = \mathbf{A}(p_0) + (p - p_0) \cdot \mathbf{u}\,\mathbf{v}^\mathsf{T}$ for a desired $p_0 \in [\underline{p}, \overline{p}]$. If the matrix $\mathbf{A}(p)$ is invertible for all p in range, the conditions for Equation 4 are perfectly met. Hence, for all $p \in [\underline{p}, \overline{p}]$, we have

$$\mathbf{A}(p)^{-1}\mathbf{b} = \mathbf{A}_0^{-1}\,\mathbf{b} - f(p)\,\mathbf{A}(p_0)^{-1}\mathbf{u}\,\mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\,\mathbf{b}\,, \tag{5}$$

$$\text{with } f(p) = \frac{p - p_0}{1 + (p - p_0) \cdot \mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\mathbf{u}}\,, \tag{6}$$

whereas the latter has valuable properties. If its denominator is nonzero, then $f(p)$ is continuous. Moreover, for $p \neq p_0$, it becomes $1/\left(1/(p - p_0) + \mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\mathbf{u}\right)$, which is clearly monotonically increasing. Due to continuity, this also holds for $p = p_0$, and hence the components of the vector $\mathbf{A}(p)^{-1}\mathbf{b}$ are monotonic in p.

If regularity of $\mathbf{A}(\mathbf{p})$ can be established for multi-valued $\mathbf{p} \in [\mathbf{p}]$, monotonicity of $\mathbf{A}(\mathbf{p})^{-1}\mathbf{b}$ with respect to all components of $\mathbf{p}$ can be proved by applying the Sherman-Morrison formula for each parameter independently, while the remaining ones are kept fixed. Then the interval-solution $[\mathbf{x}^*]$ is the smallest box containing $\mathbf{A}(\mathbf{c})^{-1}\mathbf{b}$ for all corner points $\mathbf{c}$ of the box $[\mathbf{p}]$, where $\operatorname{corners}([\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_m, \overline{x}_m])$ is defined as $\{(c_1, \ldots, c_m) \mid c_i = \underline{x}_i \text{ or } c_i = \overline{x}_i\}$. In addition, the following theorem shows, that $\mathbf{A}(\mathbf{p})^{-1}\mathbf{b}$ can be bounded more tightly by a convex set.

Theorem 4.2 (Convex-hull theorem for fill-in patterns) *Let $[\mathbf{p}] \in [\mathbb{R}]^{n_\mathbf{p}}$, and let $\mathbf{A}(\mathbf{p}) = \mathbf{A}_0 + \sum_{i=1}^{n_\mathbf{p}} p_i \cdot (\mathbf{u}_i \cdot \mathbf{v}_i^\mathsf{T})$ with $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^n$ be invertible for all $\mathbf{p} \in [\mathbf{p}]$.*

Then $\mathbf{A}(\mathbf{p})^{-1} \cdot \mathbf{b}$ lies in a convex set, corresponding to corners of $[\mathbf{p}]$, in particular its convex hull $\operatorname{conv}\left\{\mathbf{A}(\mathbf{p})^{-1} \cdot \mathbf{b} \,\middle|\, \mathbf{p} \in [\mathbf{p}]\right\}$ equals

$$\operatorname{conv}\left\{\mathbf{A}(\mathbf{c})^{-1} \cdot \mathbf{b} \,\middle|\, \mathbf{c} \in \operatorname{corners}([\mathbf{p}])\right\}.$$

Proof: Following, we will denote $\mathbf{x_c} := \mathbf{A(c)}^{-1} \cdot \mathbf{b}$. One only has to show, that for a given $\mathbf{p} \in [\mathbf{p}]$, the vector $\mathbf{x}_0 := \mathbf{A(p)}^{-1} \cdot \mathbf{b}$ is a convex-combination of the elements of $\{\mathbf{x_c} \,|\, \mathbf{c} \in \text{corners}\,([\mathbf{p}])\,\}$. We shall prove that the system of linear equations

$$\sum_{\mathbf{c}\in\text{corners}([\mathbf{p}])} \lambda_{\mathbf{c}}\, \mathbf{x_c} = \mathbf{x}_0\,, \qquad \sum_{\mathbf{c}\in\text{corners}([\mathbf{p}])} \lambda_{\mathbf{c}} = 1 \tag{7}$$

has a non-negative solution $(\lambda_{\mathbf{c}})_{\mathbf{c}\in\text{corners}([\mathbf{p}])}$. By Farkas's Lemma [13] it is sufficient to show the following: for each $\mathbf{q} \in \mathbb{R}^n$, and $q_0 \in \mathbb{R}$, where $\mathbf{q}^\mathsf{T} \cdot \mathbf{x_c} + q_0 \geq 0$ holds for all corners $\mathbf{c}$, the scalar value $\mathbf{q}^\mathsf{T} \cdot \mathbf{x}_0 + q_0$ must be non-negative.

Assume now, $\mathbf{q}$ and q_0 could be chosen in such a way that $\mathbf{q}^\mathsf{T} \cdot \mathbf{x_c} + q_0 \geq 0$, for all $\mathbf{c} \in \text{corners}\,([\mathbf{p}])$. Analogously to the case of Equation 5, the monotonicity argument shows, that $\mathbf{q}^\mathsf{T} \cdot \mathbf{x}_0$ is bounded by the smallest interval containing all $\{\mathbf{q}^\mathsf{T} \cdot \mathbf{x_c} \,|\, \mathbf{c} \in \text{corners}\,([\mathbf{p}])\,\}$. Therefore, there exists a vector $\mathbf{c} \in \text{corners}\,([\mathbf{p}])$ with $\mathbf{q}^\mathsf{T} \cdot \mathbf{x_c} \leq \mathbf{q}^\mathsf{T} \cdot \mathbf{x}_0$, and hence $\mathbf{q}^\mathsf{T} \cdot \mathbf{x}_0 + q_0 \geq \mathbf{q}^\mathsf{T} \cdot \mathbf{x_c} + q_0 \geq 0$ yields the desired relation. □

Note, that application of Theorem 4.2 is only possible, if regularity of $\mathbf{A(p)}$ can be established for all $\mathbf{p} \in [\mathbf{p}]$. For this purpose it sufficies again to look at the box corners:

Lemma 4.3 (Regularity test) *Let $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^n$, and the matrix $\mathbf{A}_0 \in \mathbb{R}^{n\times n}$ be invertible. Furthermore, let $[\mathbf{p}] \in [\mathbb{R}]^{n_\mathbf{p}}$. Then*

$$\mathbf{A(p)} = \mathbf{A}_0 + \sum_{i=1}^{n_\mathbf{p}} p_i \cdot (\mathbf{u}_i \cdot \mathbf{v}_i^\mathsf{T}) \tag{8}$$

is invertible for all $\mathbf{p} \in [\mathbf{p}]$, if and only if $\text{sign}(\det \mathbf{A(c)})$ is non-zero and constant for all $\mathbf{c} \in \text{corners}\,([\mathbf{p}])$.

Proof: First, only a single parameter p with pattern $\mathbf{u}\mathbf{v}^\mathsf{T}$ is considered. Examining the equivalent matrix products

$$\begin{pmatrix} 1 & 0 \\ p\,\mathbf{u} & \mathbf{A}_0 + p\,\mathbf{u}\,\mathbf{v}^\mathsf{T} \end{pmatrix} \cdot \begin{pmatrix} 1 & -\mathbf{v}^\mathsf{T} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\mathbf{v}^\mathsf{T} \\ 0 & \mathbf{A}_0 \end{pmatrix} \cdot \begin{pmatrix} 1 + p\,\mathbf{v}^\mathsf{T}\mathbf{A}_0^{-1}\mathbf{u} & 0 \\ \mathbf{A}_0^{-1} p\,\mathbf{u} & 1 \end{pmatrix} \tag{9}$$

one concludes, that the determinant of both sides yields

$$\det\left(\mathbf{A}_0 + p\,\mathbf{u}\,\mathbf{v}^\mathsf{T}\right) \cdot 1 = \det \mathbf{A}_0 \cdot \left(1 + p\,\mathbf{v}^\mathsf{T}\mathbf{A}_0^{-1}\mathbf{u}\right). \tag{10}$$

Hence, it follows for the case of multiple parameters, that $\det \mathbf{A(p)}$ is continuous and monotonic in each parameter p_i for all i. Therefore, a sign change of the determinant at the corner points can only occur if and only if it is zero somewhere in $[\mathbf{p}]$. □

Combining this fact with the previous theorem yields Algorithm 1, which processes $2^{n_\mathbf{p}}$ linear systems in order to treat all corners. The advantage of this procedure is, that it does not need interval computations. Of course, interval arithmetic can be used to bound rounding errors. In the case that these do not have to be tracked, existing numerical solvers, like those of analog circuit simulators [4, 12], can be utilized for tolerance analysis.

Algorithm 1 Real-valued linear system solver

Input: $\mathbf{A(p)} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $[\mathbf{p}] \in [\mathbb{R}]^{n_\mathbf{p}}$ (in the form of Theorem 4.2)

Output: $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_{2^{n_\mathbf{p}}}\}$ s. th. $\mathbf{A(p)}^{-1}\,\mathbf{b} \in \text{conv}(S)$, for all $\mathbf{p} \in [\mathbf{p}]$

```
select q ∈ [p]
set s := sign(det A(q))
if s = 0 then
   return failed
set S := ∅
for c ∈ corners([p]) do
   if sign(det A(c)) ≠ s then
      return failed
   else
      replace S := S ∪ {A(c)^-1 · b}
return S
```

As example, we consider a simple voltage divider circuit whose circuit equations can be written as follows

$$\begin{pmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 1 \\ -\frac{1}{R_1} & \frac{1}{R_1} + \frac{1}{R_2} & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ I_{V_0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ V_0 \end{pmatrix}, \tag{11}$$

for fixed $V_0 = 1\,\text{V}$ and uncertain $R_1/1\,\Omega \in [9, 11]$, and $R_2/1\,\Omega \in [90, 110]$. Since there is a one-to-one correspondence between the endpoints of R_i and $1/R_i$, one can immediately apply Algorithm 1 to the system.

Hence, the convex set can be obtained by solving equations for four corner points. Intervals bounding the range of currents and voltages can easily be obtained by component-wise computing minimum and maximum values: this yields $V_1 = 1\,\text{V}$, as well as $V_2/1\,\text{V} \in [0.891, 0.925]$ and $I_{V_0}/1\,\text{mA} \in [-10.1, -8.27]$ for the remaining quantities.

The approach described above is suitable for small parameter numbers $n_\mathbf{p}$ only, because the interval-valued problem is put down to the solution of $2^{n_\mathbf{p}}$ real-valued linear systems. In order to treat a large number of parameters, we use a kind of intervalization of the Sherman-Morrison formula to obtain a less accurate, but faster algorithm. In the case of a single uncertain parameter an interval-valued version of the Sherman-Morrison formula can be formulated.

Theorem 4.4 (Interval-valued Sherman-Morrison) *Let* $\mathbf{A}(p) = \mathbf{A}_0 + p \cdot \mathbf{u}\,\mathbf{v}^\mathsf{T} \in \mathbb{R}^{n\times n}$ *with* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, *and* $[p] \in [\mathbb{R}]$. *Let the matrix* $\mathbf{A}(p_0)$ *be invertible for some* $p_0 \in [p]$. *Then the matrix* $\mathbf{A}(p)$ *is invertible for all* $p \in [p]$ *if and only if*

$$d([p]) = 1 + ([p] - p_0) \cdot (\mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\mathbf{u}) \not\ni 0\,. \qquad (12)$$

In this case, one can calculate for right-hand side $\mathbf{b} \in \mathbb{R}^n$ *the solution* $[\mathbf{x}^*] = \left(\mathbf{A}_0 + [p] \cdot \mathbf{u}\,\mathbf{v}^\mathsf{T}\right)^{-1}\mathbf{b}$ *as*

$$[\mathbf{x}^*] = \mathbf{A}(p_0)^{-1}\mathbf{b} - [m] \cdot \left(\mathbf{A}(p_0)^{-1}\mathbf{u}\mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\mathbf{b}\right) \quad (13)$$

with $[m] = [f(\min[p]), f(\max[p])]$, *where* $f(p)$ *is defined as* $f(p) = (p - p_0) \,/\, \left(1 + (p - p_0) \cdot \mathbf{v}^\mathsf{T}\mathbf{A}(p_0)^{-1}\mathbf{u}\right)$.

Proof: Since the condition for the application of Equation 5 is fulfilled by Equation 12 the solution $\mathbf{A}(p)^{-1}\mathbf{b}$ is in the form of Equation 13. The endpoints of $[m]$ follows from the fact, that $f(p)$ is monotonically increasing. □

For illustration, the voltage divider from above is treated again, using fixed $R_2 = 100\,\Omega$. Hence, we have

$$\begin{pmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 1 \\ -\frac{1}{R_1} & \frac{1}{R_1} + 0.01 & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ I_{V_0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad (14)$$

interpreting $\frac{1}{R_1}$ as parameter with $p \equiv \frac{1\,\Omega}{R_1}$, which lies in $[\underline{p}, \overline{p}] = 1/[9, 11] = [0.0909, 0.1112]$, the fill-in pattern and the matrix $\mathbf{A}_0$ with respect to $p_0 = 0.1$ are given by

$$\mathbf{u} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \mathbf{v} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \mathbf{A}_0 = \begin{pmatrix} 0.1 & -0.1 & 1 \\ -0.1 & 0.11 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

One immediately computes

$$\mathbf{A}_0^{-1}\mathbf{b} = \begin{pmatrix} 1 \\ 0.909 \\ -0.009 \end{pmatrix}, \quad \mathbf{A}_0^{-1}\mathbf{u} = \begin{pmatrix} 1 \\ -9.0909 \\ 0.0909 \end{pmatrix},$$

and $\mathbf{v}^\mathsf{T}\mathbf{A}_0^{-1}\mathbf{u} = 9.0909$. The system is invertible for all p, because $d([\underline{p}, \overline{p}]) = [0.917, 1.102]$ does not include zero. Hence, the factor $[m] = [-0.0099, 0.0101]$ may be calculated. Using Equation 13 we get tight bounds to the solution: for the voltages $V_1 = 1\,\mathrm{V}$, as well as $V_2/1\,\mathrm{V} \in [0.900, 0.918]$ and the current $I_{V_0}/1\,\mathrm{mA} \in [-9.2, -9.0]$.

Reintroducing the second uncertain parameter $\frac{1}{R_2}$ from the example, one can clearly set $q \equiv \frac{1\,\Omega}{R_2}$ in the interval $[q] = [0.009, 0.012]$, $q_0 = 0.01$ with fill-in pattern $\mathbf{y} \cdot \mathbf{z}^\mathsf{T}$, where both vectors $\mathbf{y}$ and $\mathbf{z}$ equal the second unit vector. For tracking the variations of $\left(\mathbf{A}_0 + p \cdot \mathbf{u}\mathbf{v}^\mathsf{T} + q \cdot \mathbf{y}\mathbf{z}^\mathsf{T}\right)^{-1}\mathbf{b}$ due to both p and q the interval-valued Sherman-Morrison formula may be applied a second time, now assuming $\mathbf{A}(p)$ to be the fixed matrix. But for this purpose one cannot continue to use a real-valued matrix $\mathbf{A}_0$, instead one has to bound all possible matrices $\mathbf{A}(p) = \mathbf{A}_0 + p \cdot \mathbf{u}\mathbf{v}^\mathsf{T}$ in range.

One also has to bound $\mathbf{A}(p)^{-1}\mathbf{y}$ for all p in range, which can be done in the same way as calculating $\mathbf{A}(p)^{-1}\mathbf{b}$ above as $\mathbf{A}([p])^{-1}\mathbf{y} = (0, [8.181, 10], [0.891, 0.9244])^\mathsf{T}$. Analogously, the value given in Equation 12 is equal to $1 + ([q] - q_0) \cdot (\mathbf{z}^\mathsf{T}\mathbf{A}([p])^{-1}\mathbf{y}) = [0.990, 1.012]$. Like in the one-dimensional case, one can compute the value $(\underline{q} - q_0) \cdot (1 + (\underline{q} - q_0) \cdot \mathbf{z}^\mathsf{T}\mathbf{A}([p])^{-1}\mathbf{y})^{-1}$ and also $(\overline{q} - q_0) \cdot (1 + (\overline{q} - q_0) \cdot \mathbf{z}^\mathsf{T}\mathbf{A}([p])^{-1}\mathbf{y})^{-1}$. The smallest interval containing both forms $[\widetilde{m}] = [-0.00092, 0.0012]$, the factor which is needed for calculating the final solution $\mathbf{A}([p])^{-1}\mathbf{b} - [\widetilde{m}] \cdot \left(\mathbf{A}([p])^{-1}\mathbf{y}\right) \cdot \left(\mathbf{z}^\mathsf{T}\mathbf{A}([p])^{-1}\mathbf{b}\right)$. It leads to $V_1 \approx 1\,\mathrm{V}$, and the ranges $V_2/1\,\mathrm{V} \in [0.8908, 0.926]$ and $I_{V_0}/1\,\mathrm{mA} \in [-10.2, -8.24]$. These are slightly larger than the intervals obtained above. The reason for this is, that the parameter interval $[p]$ occurs more than once during the evaluation procedure, because $\mathbf{A}([p])^{-1}\mathbf{b}$ as well as $\mathbf{A}([p])^{-1}\mathbf{y}$ depend on it.

This approach is extented to $n_\mathbf{p}$ parameters in Algorithm 2. The number of evaluations of Formula 13 is of order $\mathcal{O}(n_\mathbf{p}^2)$. Hence, it is suited to problems with a lot of parameters, which is an advantage over Algorithm 1. But the lower complexity is payed back by reduced accuracy, because interdependencies are not removed completely.

Algorithm 2 Quick real-valued linear system solver

Input: $\mathbf{A}(p_1, \ldots, p_{n_\mathbf{p}}) = \mathbf{A}_0 + \sum_{i=1}^{n_\mathbf{p}} p_i \cdot (\mathbf{u}_i \cdot \mathbf{v}_i^\mathsf{T}) \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $\left([p_i], \ldots, [p_{n_\mathbf{p}}]\right) \in [\mathbb{R}]^{n_\mathbf{p}}$

Output: $[\mathbf{x}] \supseteq \left\{\mathbf{A}(p_1, \ldots, p_{n_\mathbf{p}})^{-1}\,\mathbf{b} \,\middle|\, p_i \in [p_i] \text{ for all } i\right\}$

for $i := 1, \ldots, n_\mathbf{p}$ **do**
  select $q_i \in [p_i]$
  set $[\tilde{p}_i] = [p_i] - q_i$
if $\det \mathbf{A}(q_1, \ldots, q_{n_\mathbf{p}}) = 0$ **then**
  return failed
set $S := \{[1,1] \cdot \mathbf{A}(q_1, \ldots, q_{n_\mathbf{p}})^{-1} \cdot \mathbf{u}_i \mid i = 1, \ldots, n_\mathbf{p}\}$
set $[\mathbf{x}] := [1,1] \cdot \mathbf{A}(q_1, \ldots, q_{n_\mathbf{p}})^{-1} \cdot \mathbf{b}$
for $[\mathbf{u}] \in S$ **do**
  replace $S := S \setminus \{[\mathbf{u}]\}$
  / $\star$ Regularity test $\star$ /
  if $1 + [\tilde{p}_i] \cdot (\mathbf{v}_i^\mathsf{T} \cdot [\mathbf{u}]) \not\ni 0$ **then**
    set $\underline{m} := \min[\tilde{p}_i] / \left(1 + \min[\tilde{p}_i] \cdot \max \mathbf{v}_i^\mathsf{T} \cdot [\mathbf{u}]\right)$
    set $\overline{m} := \max[\tilde{p}_i] / \left(1 + \max[\tilde{p}_i] \cdot \min \mathbf{v}_i^\mathsf{T} \cdot [\mathbf{u}]\right)$
    replace $S := \{[\mathbf{s}] - [\underline{m}, \overline{m}] \cdot [\mathbf{u}] \cdot (\mathbf{v}_i^\mathsf{T} \cdot [\mathbf{s}]) \mid [\mathbf{s}] \in S\}$
    replace $[\mathbf{x}] := [\mathbf{x}] - [\underline{m}, \overline{m}] \cdot [\mathbf{u}] \cdot (\mathbf{v}_i^\mathsf{T} \cdot [\mathbf{x}])$
  else
    return failed
return $[\mathbf{x}]$

4.2 Frequency-response Analysis

The small-signal analysis of analog circuits can be achieved by solving of complex-valued linear systems. The matrix also emerges from superposition of fill-in patterns, which can be defined via real vectors, but instead of real-valued parameters like resistances or conductances, the parameters are typically of the form $C \cdot s$ (capacitance) or $\frac{1}{L \cdot s}$ (inductance), for a complex-valued Laplace frequency $s = 2\,\pi\,i\,f$, and uncertain values C resp. L.

For a fixed frequency f this results in a linear complex-valued system $\mathbf{C} \cdot \mathbf{y} = \mathbf{d}$, with $\mathbf{C} \in \mathbb{C}^{n \times n}$, $\mathbf{d} \in \mathbb{C}^n$ and variables $\mathbf{y} \in \mathbb{C}^n$. In order to apply the (real) interval techniques from above it is necessary to reformulate the complex-valued equation system by the equivalent real representation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ with

$$\mathbf{A} = \begin{pmatrix} \mathrm{Re}\,\mathbf{C} & -\mathrm{Im}\,\mathbf{C} \\ \mathrm{Im}\,\mathbf{C} & \mathrm{Re}\,\mathbf{C} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} \mathrm{Re}\,\mathbf{y} \\ \mathrm{Im}\,\mathbf{y} \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathrm{Re}\,\mathbf{d} \\ \mathrm{Im}\,\mathbf{d} \end{pmatrix}.$$

In the real representation, matrix elements corresponding to the same parameter are spread over lower and upper parts of the system. Hence, the structure cannot be captured with a single fill-in pattern form only, but two of them will do.

$$\mathbf{A}(\mathbf{p}) = \mathbf{A}_0 + \sum_{\nu=1}^{n_\mathbf{p}} p_\nu \cdot (\mathbf{u}_{\mathrm{up},\nu} \cdot \mathbf{v}_{\mathrm{up},\nu}^\mathsf{T} + \mathbf{u}_{\mathrm{low},\nu} \cdot \mathbf{v}_{\mathrm{low},\nu}^\mathsf{T}) \quad (15)$$

The pair of vectors $\mathbf{u}_{\mathrm{low},\nu}, \mathbf{v}_{\mathrm{low},\nu}$ on the one hand, and $\mathbf{u}_{\mathrm{up},\nu}, \mathbf{v}_{\mathrm{up},\nu}$ on the other, share the property, that

$$\mathbf{u}_{\mathrm{low},\nu} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \mathbf{u}_{\mathrm{up},\nu} \text{ and } \mathbf{v}_{\mathrm{low},\nu} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \mathbf{v}_{\mathrm{up},\nu}. \quad (16)$$

Now, the methods developed for resistive systems above can be applied, if all occuring fill-in patterns are treated independently. For this purpose, new parameters $p_{\mathrm{up},\nu}, p_{\mathrm{low},\nu}$ are introduced, such that $\mathbf{A}(\mathbf{p})$ of Equation 15 becomes

$$\mathbf{A}_0 + \sum_{\nu=1}^{n_\mathbf{p}} p_{\mathrm{up},\nu} \cdot (\mathbf{u}_{\mathrm{up},\nu} \cdot \mathbf{v}_{\mathrm{up},\nu}^\mathsf{T}) + \sum_{\nu=1}^{n_\mathbf{p}} p_{\mathrm{low},\nu} \cdot (\mathbf{u}_{\mathrm{low},\nu} \cdot \mathbf{v}_{\mathrm{low},\nu}^\mathsf{T}).$$

Since, the dependency between lower and upper part of $\mathbf{A}(\mathbf{p})$ is lost, this approach leads to an overestimation of the range as illustrated in Figure 1.

Exploiting the structure of corresponding fill-in patterns, can be utilized to obtain the tighter wrapping of the solution set, shown by the dashed triangle in Figure 1. In order to give explicit fomulae for the triangle's corners, we first have to show some lemmas for the case of one parameter p. In order to simplify notation, one may assume temporarily that p varies in the interval $[0, \overline{p}]$, which may be obtained by suitable choice of the matrix $\mathbf{A}_0$ in Equation 15.

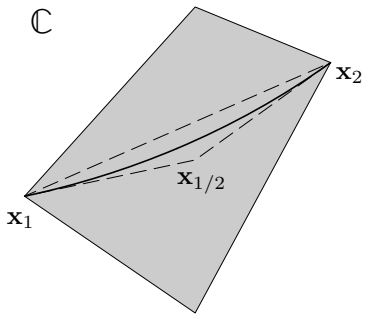


Figure 1. Parameter variance. Real representation (quadrangle), tight wrapping (dashed); $\mathbf{x}_1$, $\mathbf{x}_{1/2}$, and $\mathbf{x}_2$ correspond to $\mathbf{A}(\underline{p}, \underline{p})^{-1}$, $\frac{1}{2}(\mathbf{A}(\underline{p}, \overline{p})^{-1} + \mathbf{A}(\overline{p}, \underline{p})^{-1})$, and $\mathbf{A}(\overline{p}, \overline{p})^{-1}$, resp.

Lemma 4.5 *Let $p_0 \in \mathbb{R}$, and let $\mathbf{A} \in \mathbb{R}^{2\times 2}$ be a matrix such that $\mathbf{A} + p \cdot 1$ be invertible for all $p \geq p_0 \in \mathbb{R}$. Furthermore, assume that*

$$\mathbf{A} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}.$$

Then there exist $\lambda_1, \lambda_2 \geq 0$, such that the equality $(\mathbf{A} + p \cdot 1)^{-1} = \lambda_1 1 + \lambda_2 \cdot (\mathbf{A} + p_0 \cdot 1)^{-1}$ holds.

Proof: Since $\mathbf{A} + p \cdot 1$ is invertible, $\det(\mathbf{A} + p \cdot 1) = (a+p)^2 + b^2 \neq 0$. Hence, Cramer's rule yields

$$\begin{aligned} (\mathbf{A} + p \cdot 1)^{-1} &= \tfrac{1}{(a+p)^2+b^2} \cdot (\mathbf{A}^\mathsf{T} + p \cdot 1) \quad = \\ &\tfrac{1}{(a+p)^2+b^2} \, (\mathbf{A}^\mathsf{T} + p_0 \cdot 1 + (p - p_0) \cdot 1) \quad = \\ &\tfrac{(a+p_0)^2+b^2}{(a+p)^2+b^2} \cdot (\mathbf{A} + p_0 \cdot 1)^{-1} + \tfrac{p-p_0}{(a+p)^2+b^2} \cdot 1. \end{aligned}$$

Positivity of the squares and $p \geq p_0$ concludes the proof. □

This auxiliary result is used to prove the next lemma.

Lemma 4.6 *Let $\mathbf{A} \in \mathbb{R}^{2n\times 2n}$ be invertible, $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{2\times 2n}$ and also let $0 \leq p_0 \in \mathbb{R}$ such that $\mathbf{A} + p \cdot \mathbf{U}\,\mathbf{V}^\mathsf{T}$ is invertible for all $0 \leq p \leq p_0$. Furthermore, assume that both $\mathbf{A}, \mathbf{U}$, and $\mathbf{V}$ are of the form*

$$\begin{pmatrix} \mathbf{A}_1 & -\mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix}, \begin{pmatrix} \mathbf{u}_1 & -\mathbf{u}_2 \\ \mathbf{u}_2 & \mathbf{u}_1 \end{pmatrix}, \text{ and } \begin{pmatrix} \mathbf{v}_1 & -\mathbf{v}_2 \\ \mathbf{v}_2 & \mathbf{v}_1 \end{pmatrix},$$

respectively, for $\mathbf{A}_i \in \mathbb{R}^{n\times n}$, and $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^n$. Then there exist $\lambda_1, \lambda_2 \geq 0$ such that

$$\begin{aligned} &(\mathbf{A} + p \cdot \mathbf{U}\,\mathbf{V}^\mathsf{T})^{-1} = \\ &\mathbf{A}^{-1} - \lambda_1 \mathbf{A}^{-1}\mathbf{U}\,\mathbf{V}^\mathsf{T}\mathbf{A}^{-1} \\ &\qquad - \lambda_2 \cdot \left(\mathbf{A}^{-1} - (\mathbf{A} + p_0 \cdot \mathbf{U}\,\mathbf{V}^\mathsf{T})^{-1}\right). \end{aligned} \quad (17)$$

Proof: One may assume, that p, p_0 are strictly positive. The Sherman-Morrison-Woodbury theorem – a generalization of Theorem 4.1 – proves that $(\mathbf{A} + p \cdot \mathbf{U}\,\mathbf{V}^{\intercal})^{-1}$ is

$$\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(p^{-1}1 + \mathbf{V}^{\intercal}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}^{\intercal}\mathbf{A}^{-1}. \quad (18)$$

For the matrix $\mathbf{V}^{\intercal}\mathbf{A}^{-1}\mathbf{U}$, and $1/p \geq 1/p_0$ Lemma 4.5 shows that there exist $\lambda_1, \lambda_2 \geq 0$ with

$$\left(p^{-1}1 + \mathbf{V}^{\intercal}\mathbf{A}^{-1}\mathbf{U}\right)^{-1} = \lambda_1 1 + \lambda_2 \cdot \left(p_0^{-1}1 + \mathbf{V}^{\intercal}\mathbf{A}^{-1}\mathbf{U}\right)^{-1},$$

which can be applied to Equation 18. We obtain

$$\begin{aligned}&(\mathbf{A} + p \cdot \mathbf{U}\,\mathbf{V}^{\intercal})^{-1} = \\ &\mathbf{A}^{-1} - \lambda_1\mathbf{A}^{-1}\mathbf{U}\,\mathbf{V}^{\intercal}\mathbf{A}^{-1} \\ &\qquad - \lambda_2\mathbf{A}^{-1}\mathbf{U}\left(p_0^{-1}1 + \mathbf{V}^{\intercal}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}^{\intercal}\mathbf{A}^{-1},\end{aligned} \quad (19)$$

that equals Equation 17 again by Sherman-Morrison-Woodbury. □

Lemma 4.7 *Let $\mathbf{u}_{\mathrm{low}}, \mathbf{v}_{\mathrm{low}}, \mathbf{u}_{\mathrm{up}}, \mathbf{v}_{\mathrm{up}} \in \mathbb{R}^{2\,n}$ be vectors, such that Equation 16 holds. Furthermore, let $\mathbf{A}_0 \in \mathbb{R}^{2\,n\times 2\,n}$ be an invertible matrix arising from a real representation.*

If at least one of $\mathbf{A}_0 + \overline{p} \cdot \mathbf{u}_{\mathrm{low}} \cdot \mathbf{v}_{\mathrm{low}}^{\intercal}$ or $\mathbf{A}_0 + \overline{p} \cdot \mathbf{u}_{\mathrm{up}} \cdot \mathbf{v}_{\mathrm{up}}^{\intercal}$ is invertible for $\overline{p} \geq 0$, then there exist $\lambda \in \mathbb{R}$ such that

$$\begin{aligned}&\left(\mathbf{A}_0 + \overline{p}\,\mathbf{u}_{\mathrm{low}}\mathbf{v}_{\mathrm{low}}^{\intercal}\right)^{-1} + \left(\mathbf{A}_0 + \overline{p}\,\mathbf{u}_{\mathrm{up}}\mathbf{v}_{\mathrm{up}}^{\intercal}\right)^{-1} = \\ &2 \cdot \left(\mathbf{A}_0^{-1} - \lambda\mathbf{A}_0^{-1}\left(\mathbf{u}_{\mathrm{low}}\mathbf{v}_{\mathrm{low}}^{\intercal} + \mathbf{u}_{\mathrm{up}}\mathbf{v}_{\mathrm{up}}^{\intercal}\right)\mathbf{A}_0^{-1}\right)\end{aligned} \quad (20)$$

In case that the regularity can be established for all values $p \in [0, \overline{p}]$ with $\overline{p} > 0$, then λ is strictly positive.

Proof: First, note that the condition of Equation 16 immediately yields $\mathbf{v}_{\mathrm{low}}^{\intercal}\mathbf{A}_0^{-1}\mathbf{u}_{\mathrm{low}} = \mathbf{v}_{\mathrm{up}}^{\intercal}\mathbf{A}_0^{-1}\mathbf{u}_{\mathrm{up}}$. Using

$$\lambda = \frac{1}{2}\frac{\overline{p}}{1 + \overline{p}\,\mathbf{v}_{\mathrm{low}}^{\intercal}\mathbf{A}_0^{-1}\mathbf{u}_{\mathrm{low}}} = \frac{1}{2}\frac{\overline{p}}{1 + \overline{p}\,\mathbf{v}_{\mathrm{up}}^{\intercal}\mathbf{A}_0^{-1}\mathbf{u}_{\mathrm{up}}} \quad (21)$$

Equation 20 follows from application of Sherman-Morrison to each summand of the left-hand side. The positivity of λ is a consequence of the fact that the denominator must not be zero, but it contains 1. □

Under these conditions, one can reformulate Theorem 4.2 for the complex-valued case.

Theorem 4.8 *Let $\mathbf{u}_{\mathrm{low}}, \mathbf{v}_{\mathrm{low}}, \mathbf{u}_{\mathrm{up}}, \mathbf{v}_{\mathrm{up}} \in \mathbb{R}^{2\,n}$ be vectors, such that the conditions of Equation 16 hold. Furthermore, let $\mathbf{A}_0 \in \mathbb{R}^{2\,n\times 2\,n}$ be a matrix arising from a real representation.*

If $\mathbf{A}(p_{\mathrm{low}}, p_{\mathrm{up}}) = \mathbf{A}_0 + p_{\mathrm{low}} \cdot \mathbf{u}_{\mathrm{low}} \cdot \mathbf{v}_{\mathrm{low}}^{\intercal} + p_{\mathrm{up}} \cdot \mathbf{u}_{\mathrm{up}} \cdot \mathbf{v}_{\mathrm{up}}^{\intercal}$ is invertible for all $p_{\mathrm{low}} \in [\underline{p}, \overline{p}]$ and $p_{\mathrm{up}} \in [\underline{p}, \overline{p}]$, then the inverse $\mathbf{A}(p,p)^{-1}$ varies in

$$\mathrm{conv}\left(\mathbf{A}(\underline{p},\underline{p})^{-1}, \mathbf{A}(\overline{p},\overline{p})^{-1}, \frac{1}{2}\left(\mathbf{A}(\underline{p},\overline{p})^{-1} + \mathbf{A}(\overline{p},\underline{p})^{-1}\right)\right).$$

Proof: First of all, note that $\mathbf{A}(p_{\mathrm{low}}, p_{\mathrm{up}})$ is still a matrix arising from a real representation, which is of the form

$$\begin{pmatrix} \mathbf{A}_1 & -\mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix}, \text{ for suitable matrices } \mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{n\times n}.$$

Again, one may assume that $\underline{p} = 0$ and $\overline{p} > 0$. For abbreviation define matrices $\mathbf{A}_{1/2} = \frac{1}{2}\left(\mathbf{A}(\underline{p},\overline{p})^{-1} + \mathbf{A}(\overline{p},\underline{p})^{-1}\right)$ as well as $\mathbf{A}(p)^{-1} = \mathbf{A}(p,p)^{-1}$. Since one can always find $\mathbf{U}$ and $\mathbf{V}$ fitting into the conditions of Lemma 4.6, such that $\mathbf{U}\mathbf{V}^{\intercal}$ equals $\mathbf{u}_{\mathrm{low}}\mathbf{v}_{\mathrm{low}}^{\intercal} + \mathbf{u}_{\mathrm{up}}\mathbf{v}_{\mathrm{up}}^{\intercal}$, that lemma can be combined with Lemma 4.7. Hence, there exist $\lambda_1, \lambda_2 \geq 0$ such that $\mathbf{A}(p)^{-1}$ equals

$$\mathbf{A}_0^{-1} + \lambda_1\left(\mathbf{A}_{1/2}^{-1} - \mathbf{A}_0^{-1}\right) + \lambda_2\left(\mathbf{A}(\overline{p})^{-1} - \mathbf{A}_0^{-1}\right). \quad (22)$$

On the other hand, one can swap $\mathbf{A}(\overline{p})$ and $\mathbf{A}_0$ and replace p by $p - \overline{p}$. Thus, there are non-negative μ_1, μ_2, which can be used to construct $\mathbf{A}(p)^{-1}$ alternatively as

$$\begin{aligned}\mathbf{A}(p)^{-1} = &\mathbf{A}(\overline{p})^{-1} + \mu_1\left(\mathbf{A}_{1/2}^{-1} - \mathbf{A}(\overline{p})^{-1}\right) + \\ &\mu_2\left(\mathbf{A}_0^{-1} - \mathbf{A}(\overline{p})^{-1}\right).\end{aligned} \quad (23)$$

Inspecting linear combinations of Equation 22 and 23 shows that $\mathbf{A}(p)^{-1}$ lies in the desired convex set. □

One can immediately generalize Theorem 4.8 for the case of several parameters. For illustration consider the real representation $\mathbf{A}(p_1, p_2, p_1, p_2)$ of a complex-valued linear matrix in fill-in pattern form, which is depending on $p_1 \in [\underline{p}_1, \overline{p}_1]$ and $p_2 \in [\underline{p}_2, \overline{p}_2]$. Assuming invertibility $\mathbf{A}(p_1, p_2, p_1, p_2)^{-1} \cdot \mathbf{b}$ lies in $\mathrm{conv}(\mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_{1/2})$ with

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{A}(\underline{p}_1, p_2, \underline{p}_1, p_2)^{-1} \cdot \mathbf{b} \\ \mathbf{x}_2 &= \mathbf{A}(\overline{p}_1, p_2, \overline{p}_1, p_2)^{-1} \cdot \mathbf{b} \\ \mathbf{x}_{1/2} &= \frac{1}{2}\left(\mathbf{A}(\underline{p}_1, p_2, \overline{p}_1, p_2)^{-1} + \mathbf{A}(\overline{p}_1, p_2, \underline{p}_1, p_2)^{-1}\right)\mathbf{b}.\end{aligned}$$

Invoking the procedures again with respect to p_2 for each vector $\mathbf{x} \in \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_{\frac{1}{2}}\}$, we obtain nine points, which form up the convex set containing $\mathbf{A}(p_1, p_2, p_1, p_2)^{-1} \cdot \mathbf{b}$. The general case is illustrated in Algorithm 3.

For instance, the algorithm is exemplified by analyzing AC equations for a serial circuit. It has the same topological structure as the simple voltage divider, but one of the resistors is replaced by a capacitor. The same is true for the equations,

$$\begin{pmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 1 \\ -\frac{1}{R_1} & \frac{1}{R_1} + C_1\,s & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ I_{V_0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ V_0 \end{pmatrix}, \quad (24)$$

which is to be treated for a constant frequency of $1000\,\mathrm{Hz}$, such that we have complex-valued $s = 2\,\pi\,i \cdot 1000\,\mathrm{s}^{-1}$ with an exact independent voltage source of $V_0 = 1\,\mathrm{V}$, and two

Algorithm 3 Complex-valued linear systems solver

Input: $\mathbf{A}(\mathbf{p},\mathbf{p}) \in \mathbb{R}^{2n\times 2n}$, real representation, $\mathbf{b} \in \mathbb{R}^{2n}$, and $[\mathbf{p}] \in [\mathbb{R}]^{n_\mathbf{P}}$

Output: $R = \{\mathbf{x}_1, \ldots, \mathbf{x}_{3^{n_\mathbf{P}}}\}$ such that $\mathbf{A}(\mathbf{p},\mathbf{p})^{-1}\mathbf{b} \in \mathrm{conv}(R)$, for all $\mathbf{p} \in [\mathbf{p}]$

select $\mathbf{p}_0 \in [\mathbf{p}]$
set $s := \mathrm{sign}(\det \mathbf{A}(\mathbf{p}_0,\mathbf{p}_0))$
if $s = 0$ **then**
  return failed
/* Initialize matching sets */
set $\mathbf{S}_{[\mathbf{p}]} := \{\{(\underline{p}_1,\underline{p}_1)\},\{(\overline{p}_1,\underline{p}_1)\},\{(\underline{p}_1,\overline{p}_1),(\overline{p}_1,\underline{p}_1)\}\}$
for $i = 2,\ldots,n_\mathbf{p}$ **do**
  set $T := \emptyset$
  for $S \in \mathbf{S}_{[\mathbf{p}]}$ **do**
    $T := T \cup \left\{\{(\mathbf{p},\underline{p}_i,\mathbf{q},\underline{p}_i)\},\{(\mathbf{p},\overline{p}_i,\mathbf{q},\overline{p}_i)\} \,\middle|\, (\mathbf{p},\mathbf{q}) \in S\right\}$
    $T := T \cup \left\{\bigcup_{(\mathbf{p},\mathbf{q})\in S}\left\{(\mathbf{p},\underline{p}_i,\mathbf{q},\overline{p}_i),(\mathbf{p},\overline{p}_i,\mathbf{q},\underline{p}_i)\right\}\right\}$
  replace $\mathbf{S}_{[\mathbf{p}]} := T$
/* Start computations */
set $R := \emptyset$
for $S \in \mathbf{S}_{[\mathbf{p}]}$ **do**
  set $T := \emptyset$
  for $(\mathbf{p},\mathbf{q}) \in S$ **do**
    /* Regularity test */
    if $\det \mathbf{A}(\mathbf{p},\mathbf{q}) \neq s$ **then**
      return failed
    else
      set $T := T \cup \{(\mathbf{A}(\mathbf{p},\mathbf{q}))^{-1}\cdot\mathbf{b}\}$
  replace $R := R \cup \{\frac{1}{\#(T)}\cdot\sum_{\mathbf{x}\in T}\mathbf{x}\}$
return R

tolerance-affected parameters given as $C_1/1\,\mu\mathrm{F} \in [0.8, 1.2]$ and $R_1/1\,\Omega \in [80, 120]$. For analyzing the range using Algorithm 3, 16 linear systems have to be solved. The results can be used to make up the nine points defining the convex set (Figure 2). The smallest rectangle in the complex plane containing all possible values of the current I_{V_0} is obtained by computing the interval hull of real and imaginary parts of all points as $I_{V_0}/1\,\mathrm{mA} \in [-3.75,-1.74] + i\,[-5.52,-3.69]$.

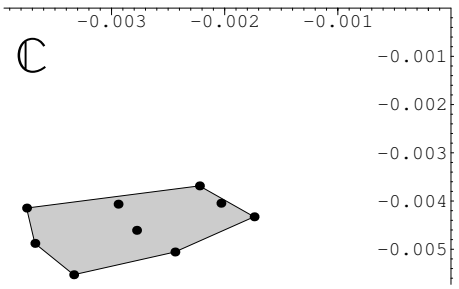


Figure 2. Range of the solution I_{V_0}.

Absolute and phase values can be estimated from the corners as $|I_{V_0}|/1\,\mathrm{mA} \in [4.3, 6.5]$, and $\varphi/1^\circ \in [-132,-112]$. This is an approximation, which gives sufficiently accurate bounds for practical applications (see also [9, Section 3.3.2]). Lower and upper bounds obtained from Algorithm 3 for the frequency response with respect to frequencies from 1 Hz up to 1 MHz can be found in Figure 3 (solid lines).

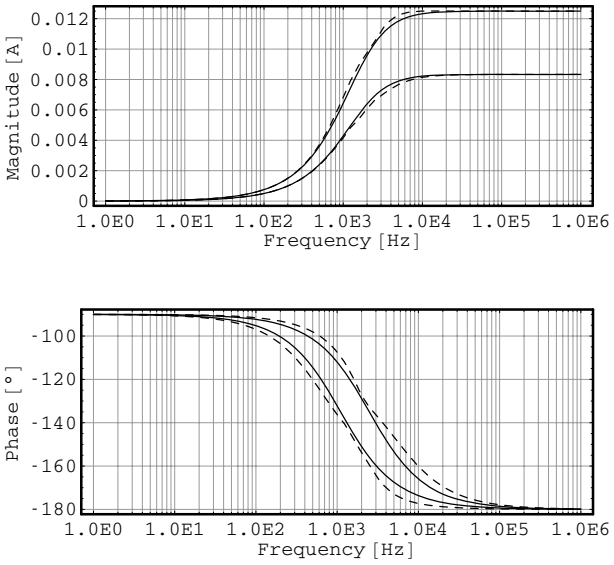


Figure 3. Outer bounds to absolute and phase value of of current I_{V_0}, obtained using Algorithm 3 (solid), faster variant (dashed).

Like in the real-valued case, the accurate approach of Algorithm 3 is not the method of choice for practical problems. For avoiding too conservative bounds, a combination with the ideas of Algorithm 2 is suitable.

For bounding of $\mathbf{A}(p,p)^{-1}\mathbf{b}$ for all $p \in [\underline{p},\overline{p}]$ the points $\mathbf{A}(\underline{p},\underline{p})^{-1}\mathbf{b}$, $\mathbf{A}(\overline{p},\overline{p})^{-1}\mathbf{b}$, $\mathbf{A}(\underline{p},\overline{p})^{-1}\mathbf{b}$, and $\mathbf{A}(\overline{p},\underline{p})^{-1}\mathbf{b}$ are needed by Theorem 4.8. Starting at $(\underline{p},\underline{p})$, an initial solution $\mathbf{A}(\underline{p},\underline{p})^{-1}\mathbf{b}$ can be computed by plain linear system solving.

In this case $1 + (\overline{p}-\underline{p})\cdot\mathbf{u}_{\mathrm{up}}^{\mathsf{T}}\,\mathbf{A}(\underline{p},\underline{p})^{-1}\,\mathbf{u}_{\mathrm{up}} > 0$ means that $1 + ([\underline{p},\overline{p}]-\underline{p})\cdot\mathbf{u}_{\mathrm{up}}^{\mathsf{T}}\,\mathbf{A}(\underline{p},\underline{p})^{-1}\,\mathbf{u}_{\mathrm{up}}$ cannot contain zero. Then Theorem 4.4 implies that $\mathbf{A}(\underline{p},p)$ is invertible for all p. Hence, the next point – namely $\mathbf{A}(\underline{p},\overline{p})^{-1}\mathbf{b}$ – can be calculated using the Sherman-Morrison formula of Equation 4. Analogously, the vector $\mathbf{A}(\overline{p},\underline{p})$ is obtained, and finally $\mathbf{A}(\overline{p},\overline{p})$ may be generated by a second application of Sherman-Morrison, like in the case of a real-valued system with two parameters.

More than one parameter can be treated, if the sets of real-valued vectors are replaced by the smallest interval-vector containing all points after each step. Therefore, from the second step on, all computations are done using interval arithmetic. Like in Algorithm 2 several auxiliary results have to be updated accordingly. Again the loss of accuracy due to wrapping intermediate results by interval vectors pays off by reducing the computational complexity to $n_{\mathbf{p}}^2$.

Back to the serial circuit example, we can now apply this procedure to Equation 24 for the same data as above, i. e. for constant frequency of $1000\,\mathrm{Hz}$, such that we have $s = 2\,\pi\,i \cdot 1000\,\mathrm{s}^{-1}$, with exact independent voltage source of $V_0 = 1\,\mathrm{V}$, and two tolerance-affected parameters $C_1/1\,\mu\mathrm{F} \in [0.8, 1.2]$ and $R_1/1\,\Omega \in [80, 120]$. Again, we have to deal with a reciprocal parameter $\frac{1}{R_1}$, whose range is then the interval $[0.0083, 0.0125]$. Since we deal with constant frequency, we can interprete $C_1\,s/i$ as new parameter varying in $[0.8\,10^{-6}, 1.2\,10^{-6}] \cdot 2\,\pi\,1000 = [0.0050, 0.0076]$. The algorithm yields bounds (in units of $1\,\mathrm{mA}$) for the current $I_{V_0} \in [-4.26, -1.74] + i\,[-5.55, -3.69]$, which properly includes $[-3.75, -1.74] + i\,[-5.52, -3.69]$, the rectangle computed using Algorithm 2.

The response for the frequencies from $1\,\mathrm{Hz}$ up to $1\,\mathrm{MHz}$ is compared to the results from Algorithm 3 in Figure 3. The curves denoting lower and upper bounds for both computations, respectively, indicate similar qualitative behavior. Although the results obtained using the faster variant are less accurate, they allow to draw conclusions about the performance of the circuit.

5 Real-world Example Application

Following, the ability to treat real-world examples is demonstrated. A prototype implementation was done as an extension to the toolbox *Analog Insydes* [4], an add-on package to the computer algebra system *Mathematica* [17], for modeling, analysis, and design of analog circuits. The operational amplifier $\mu A741$, whose schematics is illustrated in Figure 4, is considered as application. The small-signal behavior of its transistors is modeled by the simplified equiv-

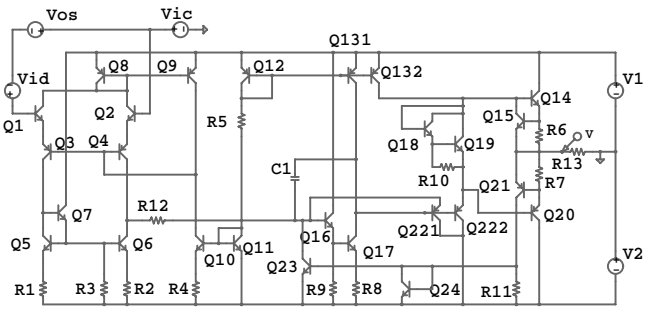


Figure 4. Operational amplifier $\mu A741$.

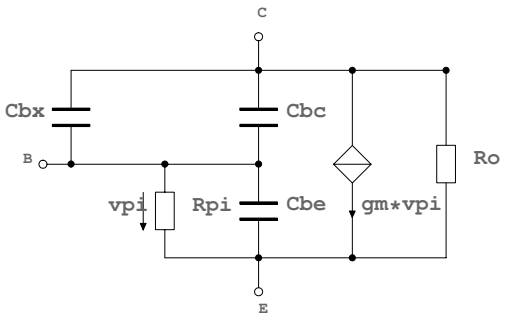


Figure 5. Simplified small-signal equivalent schematics for a bipolar-junction transistor.

alent circuit of Figure 5, which consists of resistors, capacitors, and controlled sources. The method developed in Section 4.2 is compared to Kolev's method M1 [10]: the solution $[\mathbf{x}^*]$ of frequency-response problem with real representation of the form

$$\mathbf{A}(\mathbf{p}) \cdot \mathbf{x} = \mathbf{b} \text{ with } \mathbf{A}(p_1, \ldots, p_{n_\mathbf{p}}) = \mathbf{A}_0 + \sum_{i=1}^{n_\mathbf{p}} p_i\,\mathbf{A}_i \quad (25)$$

can be bounded as $[\mathbf{x}^*] \subseteq \mathbf{x}_0 + [-1, 1] \cdot \mathbf{y}^*$, with center vector $\mathbf{x}_0 = \mathbf{A}(\mathbf{p}_0)^{-1}\mathbf{b}$, where $\mathbf{p}_0$ is the center of $[\mathbf{p}]$. The vector $\mathbf{y}^*$ is the solution of

$$\left(1 - \left|\mathbf{A}(\mathbf{p}_0)^{-1}\right| \cdot \mathbf{R}\right) \cdot \mathbf{y} = \left|\mathbf{A}(\mathbf{p}_0)^{-1} \cdot \mathbf{A}^\mathbf{P}\right| \cdot \mathbf{r}_{[\mathbf{p}]} \quad (26)$$

where $\mathbf{r}_{[\mathbf{p}]}$, whose components are defined as $(\overline{p}_i - \underline{p}_i)/2$ for each parameter range $[\underline{p}_i, \overline{p}_i]$. The remaining matrices are $\mathbf{R} = \sum_{i=1}^{n_\mathbf{p}} |\mathbf{A}_i|\ \mathbf{r}_{[\mathbf{p}]}$ and $\mathbf{A}^\mathbf{P} = \left(\mathbf{A}_1\,\mathbf{x}_0, \ldots, \mathbf{A}_{n_\mathbf{p}}\,\mathbf{x}_0\right)$. The method is applicable, if all components of $\mathbf{y}^*$ are positive.

Both approaches are applied to the linear system given in modified nodal formulation, which consists of 33 equations and 33 (complex-valued) variables. Since the small-signal behaviour is valid for constant operating point only, such circuit elements are varied, which preserve the operating point. Hence, we assign tolerances of 30% to the capacitance values.

Using the method based on successive application of the interval-valued Sherman-Morrison formula an interval-valued AC analysis is performed for 40 frequency values between $1\,\mathrm{Hz}$ and $100\,\mathrm{Hz}$ in about 370 seconds on the test system. It is slower than Kolev's M1, which finishes on the same framework within seven seconds. This is due to the fact, that overall number of computations of the latter roughly corresponds to those used for just initializing the first approach, without additional loops. Moreover, interval arithmetic is not handled efficiently in the current implementation.

The frequency response of the output voltage at node 26 is presented in Figure 6. In either case, we obtain interpretable results, but the featured method leads to tighter

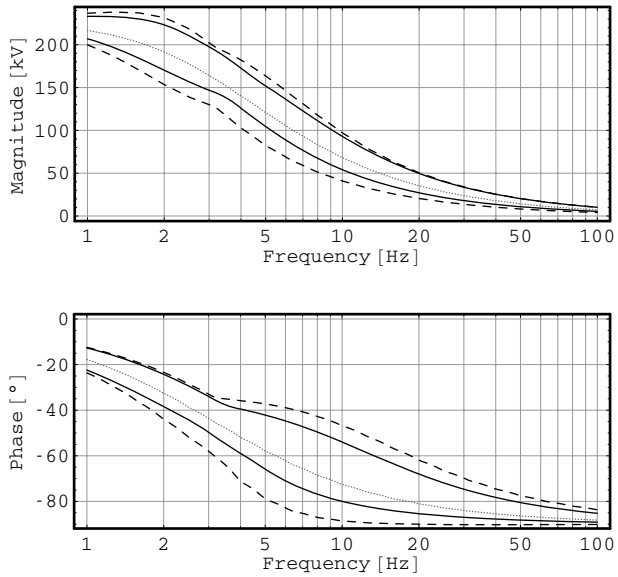


Figure 6. Frequency response bounds using the faster variant of Algorithm 3 (solid), M1 (dashed), original design point (dotted).

bounds than M1. This is mainly due to the fact, that the overestimation when successively applying the interval-valued Sherman-Morrison formula tends to be small for those temporary right-hand sides arising from fill-in pattern vectors.

6 Conclusions

Serveral techniques for obtaining outer bounds to the solution set of analog circuits with uncertain parameters have been presented. They were implemented as extension for the *Mathematica*-based electronic-design automation tool *Analog Insydes*.

The exact methods, which can be used to compute the convex hull of the region in question, are not of practical use, because of their exponential computation time. If short runtime is a crucial constraint, then Kolev's method M1 is still the prefered method for calculating meaningful outer bounds for a worst-case analysis.

The newly proposed approach, which is based of successive application of the interval-valued Sherman-Morrison formula, may be employed in case more accurate bounds are desired in acceptable time.

Acknowledgements

This work has been partially supported by the Rheinland-Pfalz cluster of excellence *Dependable Adaptive Systems and Mathematical Modelling* (DASMOD).

References

[1] A. Dreyer. Combination of symbolic and interval-numeric methods for analysis of analog circuits. In *Proc. 8th International Workshop on Symbolic Methods and Applications in Circuit Design (SMACD 2004)*, Wroclaw, Poland, Sept. 2004.

[2] A. Dreyer. *Interval Analysis of Analog Circuits with Component Tolerances*. Shaker Verlag, Aachen, Germany, 2005. Doctoral thesis.

[3] A. Dreyer. Interval methods for analog circuits. Technical Report 97, Berichte des Fraunhofer ITWM, Kaiserslautern, Germany, 2006. online available at **`http://www.itwm.fraunhofer.de`**.

[4] Fraunhofer ITWM. **`http://www.analog-insydes.de`**. Analog Insydes website.

[5] A. Ganesan, S. R. Ross, and B. R. Barmish. An extreme point result for convexity, concavity and monotonicity of parameterized linear equation solutions. *Linear Algebra an its Applications*, 390:61–73, Oct. 2004.

[6] E. R. Hansen. *Global optimization using interval methods*, volume 165 of *Monographs and textbooks in pure and applied mathematics*. Marcel Dekker, New York, 1992.

[7] E. Hennig. *Symbolic Approximation and Modeling Techniques for Analysis and Design of Analog Circuits*. Shaker Verlag, Aachen, Germany, 2000. Doctoral thesis.

[8] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*, volume 13 of *Nonconvex optimization and its applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[9] L. V. Kolev. *Interval Methods for Circuit Analysis*. World Scientific, Singapore, 1993.

[10] L. V. Kolev. Worst-case tolerance analysis of linear DC and AC electric circuits. *IEEE Transactions on Circuits and Systems*, 49(12):1–9, 2002.

[11] V. Litkovski and M. Zwolinski. *VLSI Circuit Simulation and Optimization*. Chapman & Hall, London, UK, 1997.

[12] MicroSim Corporation. *MicroSim PSpice & Basics User's Guide*, 1996.

[13] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, 1982. Papadimitriou.

[14] M. W. Tian and C.-J. R. Shi. Worst-case tolerance analysis of linear analog circuits using sensitivity bands. *IEEE Transactions on Circuits and Systems I*, 47, Aug. 2000.

[15] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design*. Van Nostrand Reinhold, New York, 2nd edition, 1993.

[16] T. Wichmann. *Symbolische Reduktionsverfahren für nichtlineare DAE-Systeme*. Berichte aus der Mathematik. Shaker Verlag, Aachen, Germany, 2004. Doctoral thesis.

[17] S. Wolfram. *The Mathematica Book*, volume 4. Wolfram Media/Cambridge University Press, 4th edition, 1999.

A Reliable Convex-Hull Algorithm for Interval-Based Hierarchical Structures

Eva Dyllong
University of Duisburg-Essen
Institute of Computer Science
Lotharstrasse 65
47048 Duisburg, Germany
dyllong@inf.uni-due.de

Abstract

This paper presents a new approach for constructing the convex polyhedral enclosure of an interval-based hierarchical structure of any dimension. To reduce the number of points in the hull construction considered, only relevant vertices on the boundary-called presumable extreme points-are involved. Additionally, a suitable update of the presumable extreme points enhances the performance whenever the maximum level of the hierarchical structure is changed. This method utilizes interval arithmetic and combines adaptation of the concept of presumable extreme points to higher dimensions with a convex-hull algorithm based on an interval linear solver.

1. Introduction

Hierarchical data structures are utilized in many practical applications. In the field of solid modeling, for example, the octree data structure provides a common technique for reconstructing a scene. This technique relies on the use of hierarchical structures of axis-aligned bounding boxes to represent objects.

There are several reasons for using an octree-based object representation. First of all, such an approach does not depend on the nature of the real solid, which is a useful property for objects with complex structures that are difficult to describe using exact mathematical expressions. The adaptive enclosure of a real solid depending only on the chosen maximum level of the tree and the efficient execution of Boolean operations are additional benefits of the hierarchical structure.

On the other hand, in most simulations the applicability of the axis-aligned octree structure to an object representation is limited. One reason for this is that, in a dynamically changing environment, a lot of arbitrary motion transformations are needed, and in general these are computationally difficult to realize for a structure based on axis-aligned bounding boxes. Whenever an object moves, its octree-based enclosure must be recalculated to reflect the new position of the object, or unaligned octrees must be used. The update of an axis-aligned octree is not as straightforward as for other object representations, such as boundary representations. On the other hand, distance computations between two objects in a scene, which are frequently used in many applications, become difficult and slow for unaligned octrees during a simulation. For this reason, several methods have been proposed to reduce the cost of such computations [11]. Using the convex hull of an octree allows one to apply algorithms for computing the distance between convex polyhedra and, by doing so, to speed up the computation.

As mentioned above, for moving objects it is advantageous to use a polyhedral instead of spatial hierarchical representation. However, the second representation is the preferred underlying data structure subsequent to the reconstruction of a scene.

In this paper, we focus on computing an adaptive and reliable polyhedral enclosure of an object at each level of the tree. This approach yields as a result a polyhedral hierarchical representation of an object. An additional benefit of the presented method is its applicability to any dimension of the hierarchical data structure. It should be noted that the resulting convex hull hierarchy is different from the bounding volume hierarchy of convex hulls which was introduced in [4] to speed up the distance calculation between complex polyhedral models and which is based on surface convex decomposition.

Our method is a generalization of the approach presented in [3] which works solely for two- or three-dimensional structures. Compared with [3], the new method does not provide different approaches for each dimension, but a unified algorithmic framework to create a reliable interval-based convex hull hierarchy of an n-tree in any dimension.

Moreover, by using interval-based hierarchical structures, we make sure that all object points are enclosed, which is an important premise in the field of motion planning, particularly in regard to collision testing. The computation of convex hull of a finite set of points is a well-known problem that was considered by several researchers [14]. Nevertheless, there are only a small number of papers regarding the reliability or accuracy of a convex hull construction, mostly dealing with the two-dimensional case [12].

2 Interval-Based Hierarchical Structure

Solid objects are generally three dimensional. But in geometry they can have any number of dimensions even though we cannot visualize objects in more than three. An n-tree data structure is a particular tree that can store an n-dimensional object. A common example is the quadtree, an established structure for storing a two-dimensional object, especially an image. Among other approaches, like boundary representation or constructive solid geometry, octrees are frequently applied to model a solid object [8].

2.1 Hierarchical Object Representation

An octree is an efficient data structure used to represent spatial data hierarchically in a tree-based structure with eight child nodes. The idea is to subdivide a cube recursively, including objects of three-dimensional space into eight mutually disjoint voxels until the required closeness to the object is achieved [13]. Each node is checked to see whether it is full (black), partially empty (gray) or empty (white) of solid material. If the nodes are empty or full, they do not need to be subdivided in further processing. In the case of partial emptiness, the nodes need to be subdivided to create a higher level of the octree. The subdivision process is repeated until all the nodes are either full or empty or until the maximum resolution level has been reached. To obtain an outer hierarchically structured approximation of the object, partially occupied leaf nodes are filled. An object representation using the octree data structure is illustrated in Figure 1.

Similar trees can be constructed for n-dimensional objects. In such cases, any gray node has to be subdivided into 2^n children nodes that correspond to the 2^n subspaces of the space occupied by the gray one.

2.2 Interval-Based Handling of the Structure

Since explicit pointer-based octree storage is more expensive in terms of memory requirements than compact linear encoding, we use depth-first (DF-) representation of

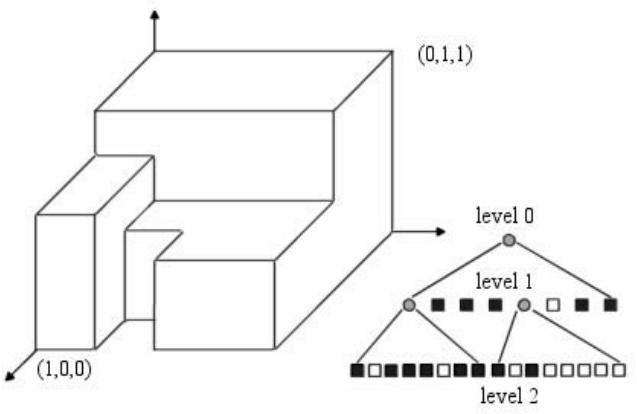


Figure 1. An example of the octree data structure.

octrees. In this representation an octree is stored by listing consecutively the octree nodes encountered when starting at the root and exploring as far as possible along each subtree before backtracking process. The symbols used are `G` (gray node), `B` (black node), and `W` (white node). Since there are only three different characters, two bits per node are sufficient for storing the octree. The octree in Figure 1, for example, yields the DF-representation `GGBWBBBWBBBBBGBWBWWWWWWBB`.

An octree node belonging to an arbitrary hierarchy level geometrically defines an axis-aligned cube. In the case of an axis-aligned octree, all cubes have as vertices machine numbers that are multiples of powers of two. But this need not be true after a rotation. To account for rounding errors, the vertices should be replaced by small intervals with machine numbers, or each side of the cube should be stored as an interval to obtain a reliable enclosure of the corresponding object. We use intervals to describe both the space occupied by a node of the spatial hierarchical structure and the space occupied by a vertex of the corresponding hierarchical polyhedral structure. Furthermore, we use interval arithmetic to carry out elementary operations on both structures.

Interval arithmetic is a common technique for providing a reliable solution to many numerical problems. In interval arithmetic numbers are replaced by intervals representing the imprecision associated with each number. Basic arithmetical operations, such as the sum, difference, product or inverse of intervals, or even elementary functions, like sine, cosine, etc., are well defined in interval arithmetic. Unfortunately, the loss of dependencies between variables is an often criticized drawback of this technique. It results in an overestimation of intervals that increases with the number of interval evaluations.

We reduce the problem of overestimation by isolating

the basic arithmetical expressions and implementing them in such a way that they yield a result that is as close as it can be to the best possible machine representation, e.g. by converting an expression to scalar product one. Since scalar products occur frequently and are important basic operations in many geometric computations, it is advantageous to perform the scalar product calculation with the same precision as the basic arithmetical operations. By using the exact scalar product, we can delay the onset of qualitative errors and improve the robustness of the implementation [2, 3].

3 The Concept of Presumable Extreme Points

The convex hull of a geometric object is the smallest convex set containing the object. In the case of a finite set of points, the hull can be identified with the smallest convex polyhedron that contains the points. The vertices of this polyhedron are called the extreme points of the convex hull.

A simple and straightforward method for determining the convex hull of an octree is to build the set of all vertices of cubes that define the octree nodes containing the object and then to construct the convex hull from these points. However, this is not an efficient method. If the maximum level of the tree changes, the points defining the vertices vary, and a completely new set of points has to be investigated. Furthermore, there are several vertices in the set under consideration that certainly belong to the interior of the object and are therefore not important for the convex hull computation.

We reduce the number of points considered by investigating a superset of the set of vertices that belong to exactly one black or one gray tree node as these points build a relevant subset of boundary points for determining the convex hull.

Proposition. The set of extreme points of the convex hull of an n-tree at level k consists of vertices that belong to exactly one black or one gray tree node of the tree at level k.

Proof. Assume, there is an extreme point e of the convex hull at level k that belongs to at least two black or gray nodes at level k. The vertices of these nodes builds a convex hull $\mathcal{H}$ which is a subset of the convex hull of the n-tree at level k since the hull considers only a subset of the tree points. The point e belongs either to the interior of the hull or to its boundary. If e is an interior point, it can not be an extreme point of either hulls. If it is on the boundary of $\mathcal{H}$ and belongs to at least two black or gray nodes, there are vertices of these nodes which lie on the same hyperplane as e because of the special architecture of an n-tree. Furthermore, e is an interior point of the section of the hyperplane generated by these vertices since solely its neighbor nodes have been picked out for the construction of $\mathcal{H}$. Thus, e can not be an extreme point of $\mathcal{H}$ and particularly of the convex hull of the n-tree. From this it follows that there is no such point e and the proof is complete.

In the following, a vertex is said to be a presumable extreme point of a tree node if none of the adjacent nodes include points of the object, or if the vertex is an update – as specified below – of such a point at a higher tree level. In further processing, we use just the presumable extreme vertices to obtain a convex polyhedral enclosure of the corresponding tree-based object representation. In addition, an adapted update of the presumable extreme vertices contributes to performance enhancement of this approach whenever the maximum level has changed.

While the maximum level of the tree increases, only gray nodes containing presumable extreme vertices take part in the update process. Figure 2 shows examples of how the update process is performed. For the sake of clarity, a two-dimensional case of a bintree has been chosen. A bintree is a data structure that partitions the underlying space by halving it recursively across the various dimensions. There is a one-to-one mapping between bintrees and n-trees.

The rectangle on the left with the black mark shows a gray node of a bintree at level k containing a presumable extreme vertex. On the opposite side of the diagram are listed the seven conceivable cases of the two children nodes at level $k + 1$. The arrows illustrate the update of the presumable extreme vertex. Both marks are kept when they are linked with an arrow and when they belong to a gray or black node after subdivision. The crossed black marks are canceled.

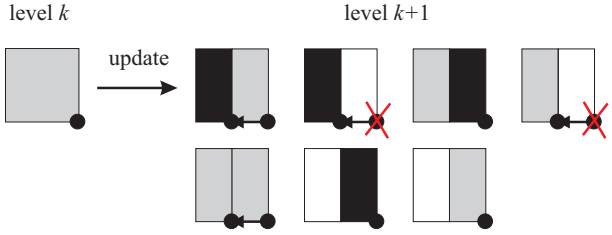


Figure 2. Update of a presumable extreme vertex.

Extreme vertices that belong to black nodes have been retained unchanged at all higher levels. The update of a gray node with a presumable extreme vertex in the upper right, upper left or bottom left corner works in an analogous manner.

The root node with black marks in each corner initiates the update processing. There are four black marks in each

corner of the root node of a quadtree, eight black marks in each corner of the root node of an octree, or generally 2^n black marks in each corner of an n-tree, we start with. We number consecutively the initialized black marks and store the corner number together with each updated black mark. Using the primary corner number allows one to make the update of a presumable extreme vertex only in some directions. For example, the update of the mark in the bottom right corner of a quadtree will only be done to the left or bottom-up directions. In the case of an n-tree at level l, we use the transformation of the n-tree into the corresponding bintree and perform the required updates of the black marks successively in n directions to obtain the set of presumable extreme vertices of the n-tree at level $l+1$.

4 Convex Polyhedral Enclosures

The convex polyhedral enclosure at each level of the tree is defined by a convex polytope. A convex polytope is the convex hull of a finite set of points. A hyperplane h supports a polytope if the polytope intersects h and lies in a closed half-space of h. The intersection of a polytope and a supporting hyperplane is called a face of the polytope. The dimension of a face is the dimension of the smallest affine space that contains the face. A k-face is a face of dimension k and is also a polytope. For a given n-dimensional polytope, its $(n-1)$-faces are called facets, its $(n-2)$-faces are called ridges, its 1-faces are edges, and its 0-faces are vertices. For a normal vector n and an offset a, a point p is beyond the corresponding hyperplane h if $\langle n, p\rangle > a$, p lies on the affine space if $\langle n, p\rangle = a$, and p is beneath h otherwise. A convex combination of a set of points is a linear combination with positive coefficients and the unit sum. The convex hull of a set of points S, $conv(S)$, is the smallest subset of S closed under convex combinations. For a more detailed introduction to the theory of convex polytopes see [17].

To compute the convex polyhedral enclosures at each level of the tree, we use a method based on a simplification of the general dimension beneath-beyond convex-hull algorithm described in [1]. In order to add a new point p to a convex hull, the incremental algorithm identifies the facets below the point. These are the visible facets to the point. The boundary of the visible facets builds the set of *horizon ridges* for the point. If there are no visible facets to point p, the point is inside the convex hull and can be discarded. Otherwise, the algorithm constructs new facets of the convex hull from horizon ridges and the processed point p and does not explicitly build the convex hulls of lower dimensional faces. A new facet of the convex hull is a facet with point p as its apex and a horizon ridge as its base. The cone of point p is the set of all new facets. The approach is based on the following theorem:

Theorem (simplified beneath-beyond) Let H be a convex polytop in $\mathcal{R}^n$ and let p be a point in $\mathcal{R}^n - H$. Then F is a facet of $conv(p \cup H)$ if and only if

1. F is a facet of H and p is below F, or
2. F is not a facet of H and its apex is p and its base is a horizon ridge of H.

Special treatment is required in the degenerate case when p lies in the affine hull of some face of H. For more details see [1].

To ensure the accuracy and correctness of the beneath-beyond algorithm it is crucial to determine in which of the two half-spaces defined by a facet of H the point p lies (the side test), and to decide whether p belongs to the affine subspace spanned by a subset of the vertices of H (the affine test for degenerate cases).

Algorithm Construction of a convex polyhedral enclosure of an n-tree at level k. The algorithm is initialized with the root node of the n-tree, which contains (presumable) extreme points in each corner of the node.

1. Update the set S of presumable extreme points from level $k-1$ of the tree.
2. Create an initial hull from a linearly independent subset of the point set S.
3. For each facet f of the hull with a non-empty outside set O, i.e. with a non-empty set of points lying above the facet f:
   - Select a furthest point q of O with respect to f.
   - Find the visible facets and horizon ridges to q.
   - Remove all visible facets to q.
   - Make a cone of new facets from q to the horizon ridges.
4. Repeat step 3 until all outside sets are empty.

In [1] a floating-point implementation of the two primitive tests may initially yield inconsistencies that are corrected afterwards. However, interval arithmetic can provide the tests in a reliable manner. To provide guaranteed results of the beneath-beyond algorithm, we utilize an interval-based linear solver as proposed in [10]. For testing the position of q with respect to f, a normal vector x of f that points to the half-space containing the initial hull has to be determined. Such a vector can be computed by solving an underdetermined linear system. A point p is beneath (beyond/in) the facet f spanned by the points $conv(p_1, \ldots, p_D)$ iff the scalar product $\langle p - p_1, x\rangle$ is larger than (smaller than/equal to) zero. In the case of interval points $[p_i]$, the interval

point $[p]$ is beneath (beyond) the facet $[f]$ iff $\inf(\langle [p] - [p_1], [x] \rangle) > 0$ ($\sup(\langle [p] - [p_1], [x] \rangle) < 0$). The side test can also be realized by determining the sign of an appropriate D-by-D determinant. If the sign of the expression $\langle p - p_1, x \rangle$ cannot be guaranteed, then we may suspect that p lies in the affine hull of the points $p_1, \ldots, p_D$. This assumption can be checked by solving the linear system

$$A^{(D)} \cdot x^{(D)} \equiv (p_1, \ldots, p_D) \cdot \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_D \end{pmatrix} = p - p_1 \quad (1)$$

with $A^{(D)} \in \mathcal{R}^{n \times D}$ and $\sum_{i=1}^{D} \xi_i = 0$. Since the points p_i are interval vectors, an enclosure $[x] = ([\xi_1], \ldots, [\xi_D])^T$ for the solution set of a linear system with an interval matrix $[A]$ on the left-hand side, and an interval vector $[p] - [p_1]$ on the right-hand side has to be determined. Given $[x]$, we can assert that p does not lie in the affine hull if $[x] = \emptyset$, or $[x] \neq \emptyset$ but $0 \notin \sum_{i=1}^{D} [\xi_i]$. If none of the assertions is fulfilled, then the system has to be resolved with rational arithmetic; for details see [10].

5. Illustrative Examples

We have implemented the hull algorithm described above in C++ programming language using the interval arithmetic of C-XSC (a C++ class library for eXtended Scientific Computation) [7]. For visualization we utilize the library OpenGL and the graphical widget toolkit Qt [16, 6].

The implementation provides an n-tree data structure, in particular several facilities for creating the octree representation of an object. A tree can be imported from an ASCII file containing the DF-representation of the tree or containing another type of hierarchical representation, such as subpavings from SIVIA (Set Inversion Via Interval Analysis) [9]. Figure 3 shows an example of an octree defined by a DF-representation in Section 2.2 and its convex hull at level 2.

Furthermore, an axis-aligned tree can be created manually node by node. An interval polynomial expression can be also put as an input. An octree at level 5 computed for the interval polynomial expression

$$x^2 - y^3 + z^2 = [0.95, 1.01] \quad (2)$$

if using our implementation is illustrated in Figure 4.

To reliably decide which parts of the scene belong to an object, a common inclusion test applying interval arithmetic is used. Alternatively, to test the sign of a multivariate polynomial in a box, an interval version of the criterion from Walach and Zeheb has been implemented [15, 5].

Constructive solid geometry trees (CSG trees) can also be used as an input to form complex objects. The following figures demonstrate the result of a transformation of CSG objects into interval-based octrees. Figures 5 and 6 illustrate a scene containing rotated interval-based octrees and their polynomial enclosures at level 7, respectively.

The tool implemented has been upgraded with accurate distance routines for interval-based hierarchical structures, or rather *separation* routines since $dist(A, B) = \min \|a - b\|_2$, $a \in A$, $b \in B$ is not a metric on the set of closed subsets in Euclidean space, as the Hausdorff distance is. Nevertheless, we call them just distance routines. Finally, the computation time required for the distance computation between two hierarchical object representations has been compared with the time needed for distance computation between their polyhedral enclosures [6]. Owing to the reduction of points that define an object, a noticeable saving in computation time was expected as a result of building its polyhedral enclosure. In the case of the octree described in Figure 1, for example, there are thirteen cube nodes each with eight vertices and twelve rectangle facets to be considered during the distance computation as opposed to eleven vertices and seventeen triangular facets in the case of its convex enclosure.

Therefore, the time reduction in the case of moving objects is all the more significant and was detected in several examples through distance computation times that were decreased by a factor often equal to ten or greater as reported in [6]. For example, for the scene with two octree-encoded parts of spheres at level 5 depicted in Figure 7 altogether 2089182 basic geometrical operations, like distance calculation between a point and a line, visibility tests etc., are needed for distance investigation compared with 156436 operations required for a scene containing the convex enclosures of the spheres. In the latter case, the running time on Intel(R) Pentium(R) M processor with 1.6 GHz is 1.59 s. After several improvements in order to performance enhancement of the distance routines for octrees moving under rigid motions that have been made, the octree-based

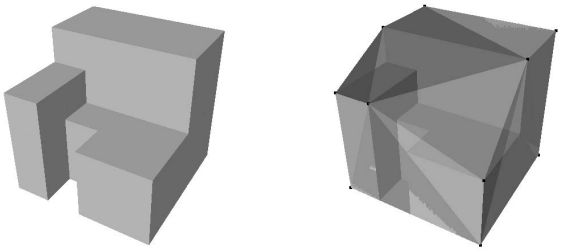


Figure 3. The octree described in Figure 1 and its convex hull at level 2.

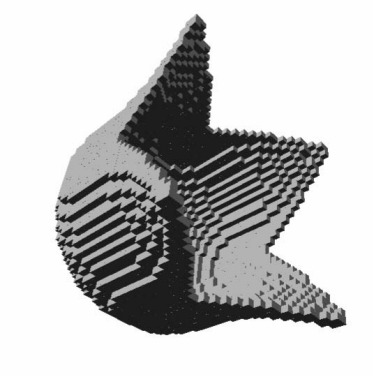


Figure 4. An octree at level 5 defined by the interval expression (2).

scene still remains 353578 operations, i.e. twice as much as in case of polyhedral enclosures.

Nevertheless, the distance algorithm for octrees computes the actual distance between two objects, while the algorithm for polyhedra computes only the lower bound obtained by enclosing the objects in their convex hulls. Hence, a comparison of our improved distance algorithm for octrees in different coordinate systems with the branch-and-bound approach described in [11] would be worthwhile and is intended.

6 Conclusions

In this paper we have presented a new algorithm for constructing a reliable hierarchical polyhedral enclosure at each level of an interval-based n-tree. The use of interval arithmetic guarantees that no part of the n-dimensional object is ever missed in the enclosure. This is an important property for many applications, including path planning. The approach utilizes an efficient update of presumable extreme points on the boundary combined with a general dimension convex hull method based on the beneath-beyond algorithm and an interval linear solver. The result is the smallest machine-representable convex polyhedral enclosure containing the n-tree representation of an object at each level of the tree. An efficient construction of a convex polyhedral hierarchical structure is an advantage for further processing. For example, it can be used to speed up distance calculations between objects in a scene.

7. Acknowledgments

This research was carried out within the scope of the recent project "Interval-based approaches for adaptive hierarchical models in modeling and simulation systems" funded by the German Research Council (DFG).

We would like to thank the anonymous referees for their helpful suggestions and corrections that improved the earlier version of this paper.

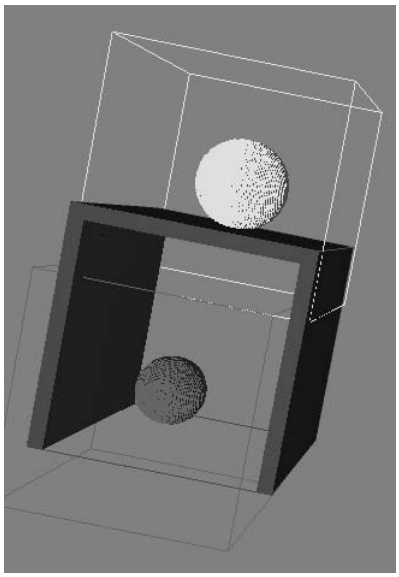


Figure 5. A scene with rotated octrees of level 7.

References

[1] Barber, C. B., Dobkin, D. P, Huhdanpaa, H.: The Quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software 22(4): 469–483, 1996.

[2] Dyllong, E.: Akkurate Abstandsalgorithmen mit Ergebnisverifikation. Ph.D. thesis, University of Duisburg-Essen, VDI Reihe 20, Nr. 390, Düsseldorf, 2004.

[3] Dyllong, E., Luther, W.: Verified convex hull and distance computation for octree-encoded objects. Journal of Computational and Applied Mathematics 199(2): 358-.364, 2006.

[4] Ehmann, S., Lin, M.C.: Accurate and fast proximity queries between polyhedra using convex surface decomposition. Computer Graphics Forum (Proc. of Eurographics2001) 20(3): 500-510, 2001.

[5] Fausten, D., Luther, W.: Verifizierte Lösungen von nichtlinearen polynomialen Gleichungssystemen. Technical Report SM-DU-477, University of Duisburg, 2000.

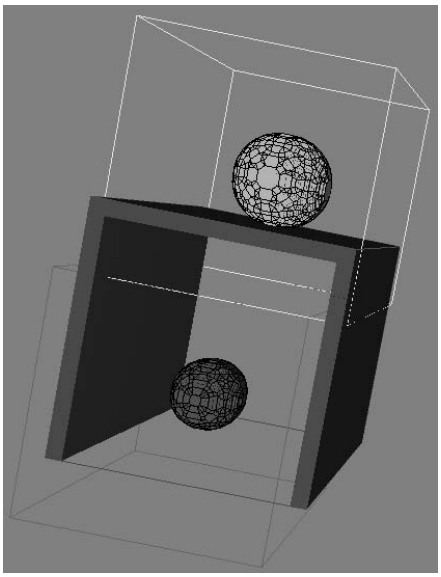


Figure 6. The polyhedral enclosures of the rotated octrees of level 7.

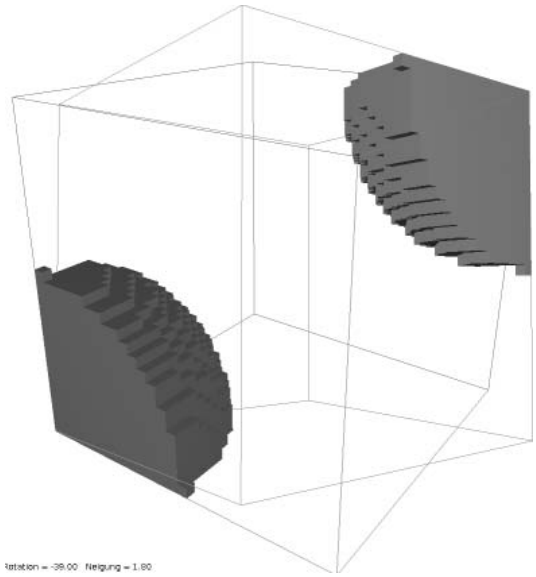


Figure 7. A scene with two octree-encoded spheres.

[6] Grimm, C.: Verläßliche Abstandsalgorithmen für intervallbasierte Octreemodelle und ihre konvexen Einschlüsse – Ein Effizienzvergleich. Diploma thesis, University of Duisburg-Essen, 2006.

[7] Hammer, R., Hocks, M., Kulisch, U., and Ratz, D.: C++ Toolbox for Verified Computing. Basic Numerical Problems. Springer, Berlin, 1995.

[8] Hoffmann, C. M.: Geometric and Solid Modeling. Morgan Kaufmann, 1989.

[9] Jaulin, J., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, London, 2001.

[10] Krivsky, S., Lang, B.: Using Interval Arithmetic for Determining the Structure of Convex Hulls. Numerical Algorithms 37(1-4): 233–240, 2004.

[11] Major, F., Malenfant, J. and Stewart, N. F.: Distance between objects represented by octtrees defined in different coordinate systems. Computers and Graphics 13(4): 497–503, 1989.

[12] Ratschek, H., Rokne, J.: Geometric computations with interval and new robust methods: applications in computer graphics, GIS and computational geometry. Horwood Publishing, Chichester, 2003.

[13] Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Company, 1990.

[14] Seidel, R.: Convex hull computations. In: Handbook of discrete and computational geometry, CRC Press LLC, Boca Raton, pp. 361–375, 1997.

[15] Walach, E., Zeheb, E.: Sign Test of Multivariate Real Polynomials. IEEE Trans. on Circuits and Systems 27(7): 619–625, 1980.

[16] Zhang, M.: Konvexe Einschlüsse von hierarchischen intervallbasierten Modellen. Diploma thesis, University of Duisburg-Essen, 2005.

[17] Ziegler, G. M.: Lectures on Polytopes. Graduate Texts in Mathematics (vol. 152). Springer, New York, 2006.

Parametric Linear System of Equations, Whose Elements are Nonlinear Functions

El-Owny, H.
Bergische Universität Wuppertal
Faculty C - Department of Mathematics and Computer Science
Scientific Computing / Software Engineering
Gaußstraße 20
42097 Wuppertal
Germany
hassan.el-owny@math.uni-wuppertal.de

Abstract

This paper addresses the problem of solving parametric linear systems of equations whose coefficients are, in the general case, nonlinear functions of interval parameters. Such systems, are encountered in many practical problems, e.g in electrical engineering and mechanical systems. A C-XSC[8] implementation of a parametric fixed-point iteration method for computing an outer enclosure for the solution set is proposed in this paper. Numerical examples illustrating the applicability of the proposed method are solved, and compared with other methods.

Keywords: parametric linear systems, validated interval software, C-XSC, nonlinear functions, Generalized Interval Arithmetic.

1. Introduction

In many practical applications [3], parametric interval systems involving uncertainties in the parameters have to be solved. In most engineering design problems, linear prediction problems, models in operation research, etc. [15] there are usually complicated dependencies between coefficients. The main reason for this dependency is that the errors in several different coefficients may be caused by the same factor [16, 11]. More precisely, consider a parametric system

$$A(p) \cdot x = b(p), \tag{1}$$

where $A(p) \in \mathbb{R}^{n\times n}$ and $b(p) \in \mathbb{R}^n$ depend on a parameter vector $p \in \mathbb{R}^m$. The elements of $A(p)$ and $b(p)$ are, in general, nonlinear functions of m parameters

$$\begin{aligned} a_{ij}(p) &= a_{ij}(p_1, \cdots, p_m), \\ b_i(p) &= b_i(p_1, \cdots, p_m), \quad (i, j = 1, \cdots, n). \end{aligned} \tag{2}$$

When p varies within a range $[p] \in I\mathbb{R}^m$, the set of solutions to all $A(p) \cdot x = b(p)$, $p \in [p]$, is called parametric solution set, and is represented by

$$\sum\nolimits^{p} := \sum(A(p), b(p), [p]) := \{x \in \mathbb{R}^n | A(p) \cdot x = b(p) \text{ for some } p \in [p]\}. \tag{3}$$

Since the solution set has a complicated structure which is difficult to find [17], one looks for the interval hull $\diamond(\sum)$ where $\sum$ is a nonempty bounded subset of $\mathbb{R}^n$. For $\sum \subseteq \mathbb{R}^n$, define $\diamond : P\mathbb{R}^n \longrightarrow I\mathbb{R}^n$ by[1]

$$\diamond(\sum) := [\inf \sum, \sup \sum] = \cap\{[x] \in I\mathbb{R}^n | \sum \subseteq [x]\}.$$

It is well-known that [17]

$$\sum(A(p), b(p), [p]) \subseteq \sum(A([p]), b([p])),$$

where

$$\begin{aligned} A([p]) &:= \diamond\{A(p) \in \mathbb{R}^{n\times n} | p \in [p]\}, \\ b([p]) &:= \diamond\{b(p) \in \mathbb{R}^n | p \in [p]\} \end{aligned}$$

are the non-parametric interval matrix, respectively vector, that correspond and are obtained from the parametric ones.

[1] $P\mathbb{R}^n$ is the power set over $\mathbb{R}^n$. Given a set S the power set of S is the set of all subset of S

Hence, $A([p])\cdot x = b([p])$ is the non-parametric system corresponding to the parametric one, and non-parametric solution set is defined by

$$\sum^{g} := \sum(A([p]), b([p])) := \{x \in \mathbb{R}^n | A \cdot x = b \text{ for some } A \in A([p]), \quad b \in b([p])\} \quad (4)$$

The calculation of $\diamond(\sum)$ is also quite expensive, so it would be a more realistic task to find an enclosure of it, this means computation of $[y] \in I\mathbb{R}^n$ such that $[y] \supseteq \diamond(\sum^p) \supseteq \sum^p$. Probably the first general purpose method computing outer (and inner) bounds for $\diamond(\sum^p)$ is based on the fixed-point interval iteration theory developed by S. Rump. In [20] Rump applies the general verification theory for system of nonlinear equations for solving parametric linear systems involving affine-linear dependencies. This method was generalized in [18] by proving that a sharp enclosure of the iteration matrix expands the scope of application of the method over problems involving the so-called column-dependent matrices. Meanwhile, there were many attempts to construct suitable methods for solving parameter dependent interval linear systems [3, 11, 15, 16, 17, 19, 9, 14]. We do not intend to give here a complete overview of methods used for solving linear systems with dependent data. Most of the methods developed so far address linear systems involving affine-linear dependencies between the parameters. Very few articles [3, 10] studied the general case where $a_{ij}(p)$ and $b_i(p)$, $(i, j = 1, \cdots, n)$ are nonlinear functions of a parameter vector p.

The goal of this paper is computing an outer solution of the parametric system (1) in the case of nonlinear functions, with the aid of a C-XSC [7, 8, 6, 12] implementation for a Generalized Interval Arithmetic , which has been proposed by Hansen in 1975 [4] (the main goal of Generalized Interval Arithmetic is to reduce the dependency problem in the interval arithmetic, in addition to enclosing ranges of nonlinear interval functions by linear interval forms).

We use the following notations. $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times n}$, $I\mathbb{R}$, $I\mathbb{R}^n$, $I\mathbb{R}^{n\times n}$, to denote the set of real numbers, the set of real vectors with n components, the set of real $n \times n$ matrices, the set of intervals, the set of intervals vectors with n components and the set of $n \times n$ intervals matrices, respectively. For a real interval $[x]$ we mean a real compact interval $[x] = [\underline{x}, \overline{x}] := \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}$, where $\underline{x}$ and $\overline{x}$ denote the lower and upper bounds of the interval $[x]$, respectively. For an interval $[x]$ define the mid-point

$$\check{x} = \text{mid}([x]) = (\underline{x} + \overline{x})/2$$

and the radius

$$\text{rad}([x]) = (\overline{x} - \underline{x})/2.$$

Definition of real intervals and operations with such intervals can be found in a number of references [1, 5, 13]. However, we present the main interval arithmetic operation. For $[x], \ [y] \in I\mathbb{R}$

$$\begin{aligned}
[x] + [y] &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\
[x] - [y] &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}], \\
[x] \cdot [y] &= [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})], \\
1/[y] &= [1/\overline{y}, 1/\underline{y}] \quad \text{if} \quad 0 \notin [y], \\
[x]/[y] &= [x] \cdot [1/\overline{y}, 1/\underline{y}] \quad \text{if} \quad 0 \notin [y].
\end{aligned}$$

2. The dependency problem

The dependency problem arises when one or several variable occur more than once in an interval expression. Dependency may lead to catastrophic overestimation in interval computations. For example, if the interval $[x] = [1, 2]$ is subtracted from itself, $[x] - [x] = [-1, 1]$ is obtained as a result. The result is not the interval $[0, 0]$ as expected. Actually, interval arithmetic can not recognize the multiple occurrence of the same variable $[x]$. The result is $\{x - y | x \in [x], y \in [x]\}$ instead of $\{x - x | x \in [x]\}$. In general, when a given variable occurs more than once in an interval computation, it is treated as a different variable in each occurrence. A Generalized Interval Arithmetic method has been developed by Hansen [4] in 1975. Its purpose is to reduce the effect of the dependency problem when computing with interval arithmetic, in addition to enclosing ranges of nonlinear interval functions by linear interval forms. In [6] we have realized this method in the environment of C-XSC.

3. Theoretical Background

In this section we give a brief summary of the theory of the enclosure method for our problem, in case of the system (1) involving affine-linear dependencies between the parameters.

Theorem 1. Popova [17] *Consider parametric linear system (1), where $A(p)$ and $b(p)$ are defined by*

$$\begin{aligned}
a_{ij}(p) &:= a_{ij}^{(0)} + \sum_{\nu=1}^{m} p_\nu a_{ij}^{(\nu)}, \\
b_i(p) &:= b_i^{(0)} + \sum_{\nu=1}^{m} p_\nu b_i^{(\nu)}, \quad (i, j = 1, 2, \cdots, n).
\end{aligned}$$

Let $R \in \mathbb{R}^{n\times n}$, $[y] \in I\mathbb{R}^n$, $\tilde{x} \in \mathbb{R}^n$ be given and define $[z] \in I\mathbb{R}^n$ and $[C(p)] \in I\mathbb{R}^{n\times n}$ by

$$[z] := R\cdot(b^{(0)} - A^{(0)}\tilde{x}) + \sum_{\nu=1}^{m}[p_\nu](Rb^{(\nu)} - RA^{(\nu)}\cdot\tilde{x}),$$

$$[C(p)] := I - R\cdot A^{(0)} - \sum_{\nu=1}^{m}[p_\nu](R\cdot A^{(\nu)}),$$

where $A^{(0)} := \left(a_{ij}^{(0)}\right), \cdots, A^{(m)} := \left(a_{ij}^{(m)}\right) \in \mathbb{R}^{n\times n}$, $b^{(0)} := \left(b_i^{(0)}\right), \cdots, b^{(m)} := \left(b_i^{(m)}\right) \in \mathbb{R}^n$. Define $[v] \in I\mathbb{R}^n$ by means of the following Single step method

$$1 \le i \le n : [v_i] = \{\diamond\{[z] + [C]\cdot[u]\}\}_i \quad \text{where}$$
$$[u] := ([v_1], \cdots, [v_{i-1}], [y_i], \cdots, [y_n])^\top.$$

If $[v] \overset{\circ}{\subset} [y]$[1], then R and every matrix $A(p)$, $p \in [p]$ is regular, and for every $p \in [p]$ the unique solution $\hat{x} = A^{-1}(p)b(p)$ of $A(p)\cdot x = b(p)$ satisfies $\hat{x} \in \tilde{x} + [v]$.

The above theorem generalizes theorem 4.8 from [20] by requiring of the range of $C(p)$ instead of using an interval extension $C([p])$ [18].

4. Main Results

In this section, a method for computing an outer solution for the system (1), in the general case, is suggested. The derivation of the method is based on the approach employed in [4, 6].

Let $f : [x] \subset \mathbb{R}^m \longrightarrow \mathbb{R}$ be a continuous function. The function $f(x)$ can be enclosed by the following linear interval form

$$[L_f(x)] := [c^f] + \sum_{\nu=1}^{m}\zeta_\nu[v_k^f], \quad x \in [x] \tag{5}$$

where $[c^f]$ and $[v_\nu^f]$, $\nu = 1, \cdots, m$ are real intervals, and $\zeta_\nu \in [-\text{rad}([x_\nu]), \text{rad}([x_\nu])]$. The form (5) can be determined in an automatic way using the algorithms of [6]. It has the inclusion property

$$f(x) \in [L_f(x)], \quad x \in [x].$$

Example 1. *Let*

$$f(x) = \frac{x_1 + x_2}{x_1 - x_2}$$

with $x_1 \in [5, 10]$, $x_2 \in [1, 2]$.

[1] $\overset{\circ}{\subset}$ is the inner inclusion relation

Using Algorithms of [6], the linear interval form (5) can be computed for the above function. It will be as follows

$$[L_f] := [1.5, 1.5] + [-0.167, -0.05]\zeta_1 + [0.277, 0.8334]\zeta_2$$

where $\zeta_1 \in [-2.5, 2.5]$, $\zeta_2 \in [-0.5, 0.5]$.

When we reduced the linear interval form $[L_f]$ to an ordinary interval, we obtain

$$\begin{aligned}\text{reduce}([L_f]) =& \text{reduce}([1.5, 1.5] + [-0.167, -0.05][-2.5, 2.5] \\ &+ [0.277, 0.8334][-0.5, 0.5]) \\ =& [0.666, 2.334].\end{aligned}$$

We assume that $a_{ij}(p)$ and $b_i(p)$, $i, j = 1, \cdots, n$ in (2) are continuous functions. In accordance with (5), the corresponding linear interval forms are

$$[L_{ij}(p)] := [c^{a_{ij}}] + \sum_{\nu=1}^{m}\zeta_\nu[v_\nu^{a_{ij}}] \ni a_{ij}(p) \tag{6}$$

$$[l_i(p)] := [c^{b_i}] + \sum_{\nu=1}^{m}\zeta_\nu[v_\nu^{b_i}] \ni b_i(p), \tag{7}$$

where $\zeta_\nu \in [-\text{rad}([p_\nu]), \text{rad}([p_\nu])]$, $\nu = 1, \cdots, m$.

According to the above two relations, we introduce the $m + 1$ numerical interval matrices

$$\begin{aligned}\left[\mathcal{A}^{(0)}\right] &:= \left([c^{a_{ij}}]\right), \quad \left[\mathcal{A}^{(1)}\right] := \left([v_1^{a_{ij}}]\right), \cdots\cdots, \\ \left[\mathcal{A}^{(m)}\right] &:= \left([v_m^{a_{ij}}]\right) \in I\mathbb{R}^{n\times n}\end{aligned} \tag{8}$$

and the corresponding numerical interval vectors

$$\begin{aligned}\left[\ell^{(0)}\right] &:= \left([c^{b_i}]\right), \quad \left[\ell^{(1)}\right] := \left([v_1^{b_i}]\right), \cdots\cdots, \\ \left[\ell^{(m)}\right] &:= \left([v_m^{b_i}]\right) \in I\mathbb{R}^n.\end{aligned} \tag{9}$$

Hence, a new parametric interval matrix and a right-hand side parametric interval vector can be represented by

$$\left.\begin{aligned}[\mathcal{A}(\zeta)] &= [\mathcal{A}^{(0)}] + \textstyle\sum_{\nu=1}^{m}\zeta_\nu[\mathcal{A}^{(\nu)}], \\ [\ell(\zeta)] &= [\ell^{(0)}] + \textstyle\sum_{\nu=1}^{m}\zeta_\nu[\ell^{(\nu)}].\end{aligned}\right\} \tag{10}$$

According to the parametric system (1), where its elements are defined by (2). We can write a new parametric interval system in the following form

$$\begin{aligned}[\mathcal{A}(\zeta)]\cdot x &= [\ell(\zeta)], \\ \left([\mathcal{A}^{(0)}] + \sum_{\nu=1}^{m}\zeta_\nu[\mathcal{A}^{(\nu)}]\right)\cdot x &= [\ell^{(0)}] + \sum_{\nu=1}^{m}\zeta_\nu[\ell^{(\nu)}],\end{aligned} \tag{11}$$

where the new parametric vector ζ varies within the range $[\zeta] \in I\mathbb{R}^m$.

Example 2. *Let*

$$\begin{pmatrix} -(p_1+p_2)p_2 & p_2p_4 \\ p_4p_5 & p_3p_5 \end{pmatrix} \cdot x = \begin{pmatrix} p_1p_2 \\ p_2p_3 \end{pmatrix}, \qquad (12)$$

with $[p] = ([0.96, 0.98], [1.92, 1.96], [0.96, 0.98], [0.48, 0.5], [0.48, 0.5])^T \in I\mathbb{R}^5$.

According to Algorithms presented in [6], the linear interval forms (6) and (7) can be computed for every element of the matrix and the right hand side of the system (12), respectively.

$$\begin{aligned}
[L_{11}(p)] &:= [-5.6559, -5.6453] + [-1.9601, -1.9199]\zeta_1 \\
&\quad + [-4.8501, -4.8499]\zeta_2 \\
[L_{12}(p)] &:= [0.95059, 0.95061] + [0.47999, 0.5]\zeta_2 \\
&\quad + [1.9399, 1.94]\zeta_4 \\
[L_{21}(p)] &:= [0.24009, 0.24011] + [0.47999, 0.5]\zeta_4 \\
&\quad + [0.4899, 0.49]\zeta_5 \\
[L_{22}(p)] &:= [0.47529, 0.47531] + [0.47999, 0.5]\zeta_3 \\
&\quad + [0.96999, 0.97]\zeta_5 \\
[l_1(p)] &:= [1.8817, 1.8818] + [1.9199, 1.9601]\zeta_1 \\
&\quad + [0.96999, 0.97]\zeta_2 \\
[l_2(p)] &:= [1.8817, 1.8818] + [0.95999, 0.98001]\zeta_2 \\
&\quad + [1.9399, 1.94]\zeta_3
\end{aligned}$$

where $\zeta_1 \in [-0.01, 0.01]$, $\zeta_2 \in [-0.02, 0.02]$, $\zeta_3 \in [-0.01, 0.01]$, $\zeta_4 \in [-0.01, 0.01]$ *and* $\zeta_5 \in [-0.01, 0.01]$.

From (8), the 6 *numerical interval matrices can be defined as follows*

$$\begin{aligned}
[A^{(0)}] &:= \begin{pmatrix} [-5.6559, -5.6453] & [0.9505, 0.95061] \\ [0.24009, 0.24011] & [0.47529, 0.47531] \end{pmatrix}, \\
\left[A^{(1)}\right] &:= \begin{pmatrix} [-1.9601, -1.9199] & [0, 0] \\ [0, 0] & [0, 0] \end{pmatrix}, \\
\left[A^{(2)}\right] &:= \begin{pmatrix} [-4.8501, -4.8499] & [0.4799, 0.5] \\ [0, 0] & [0, 0] \end{pmatrix}, \\
\left[A^{(3)}\right] &:= \begin{pmatrix} [0, 0] & [0, 0] \\ [0, 0] & [0.4799, 0.5] \end{pmatrix}, \\
\left[A^{(4)}\right] &:= \begin{pmatrix} [0, 0] & [1.9399, 1.94] \\ [0.4799, 0.5] & [0, 0] \end{pmatrix}, \\
\left[A^{(5)}\right] &:= \begin{pmatrix} [0, 0] & [0, 0] \\ [0.4899, 0.49] & [0.9699, 0.97] \end{pmatrix},
\end{aligned}$$

and from (9), the corresponding numerical interval vectors can be defined as follows

$$\begin{aligned}
[\ell^{(0)}] &:= \begin{pmatrix} [1.8817, 1.8818] \\ [1.8817, 1.8818] \end{pmatrix}, \\
\left[\ell^{(1)}\right] &:= \begin{pmatrix} [1.9199, 1.9601] \\ [0, 0] \end{pmatrix}, \\
\left[\ell^{(2)}\right] &:= \begin{pmatrix} [0.9699, 0.97] \\ [0.9599, 0.98001] \end{pmatrix}, \\
\left[\ell^{(3)}\right] &:= \begin{pmatrix} [0, 0] \\ [1.9399, 1.94] \end{pmatrix}, \\
\left[\ell^{(4)}\right] = \left[\ell^{(5)}\right] &:= \begin{pmatrix} [0, 0] \\ [0, 0] \end{pmatrix}.
\end{aligned}$$

We get a new parametric interval system by substituting the above numerical interval matrices and the corresponding numerical interval vectors in (11).

The following theorem is a modification of theorem 1.

Theorem 2. *Consider the parametric linear system (1), where* $A(p)$ *and* $b(p)$ *are given by (2). Let* $[\mathcal{A}(\zeta)] \in I\mathbb{R}^{n\times n}$ *and* $[\ell(\zeta)] \in I\mathbb{R}^n$ *be given by (10) with* $\zeta \in \mathbb{R}^m$, *and let* $R \in \mathbb{R}^{n\times n}$, $[y] \in I\mathbb{R}^n$, $\tilde{x} \in \mathbb{R}^n$ *be given and define* $[z] \in I\mathbb{R}^n$ *and* $[C(\zeta)] \in I\mathbb{R}^{n\times n}$ *by*

$$\begin{aligned}
[z] &:= R \cdot ([\ell^{(0)}] - [\mathcal{A}^{(0)}] \cdot \tilde{x}) + \\
&\quad \sum_{\nu=1}^{m} R \cdot ([\ell^{(\nu)}] - [\mathcal{A}^{(\nu)}] \cdot \tilde{x})[\zeta_\nu], \\
[C(\zeta)] &:= I - R \cdot [\mathcal{A}^{(0)}] - \sum_{\nu=1}^{m} (R \cdot [\mathcal{A}^{(\nu)}])[\zeta_\nu].
\end{aligned}$$

Define $[v] \in I\mathbb{R}^n$ *by means of the following Single step method*

$$1 \le i \le n : [v_i] = \{\diamond\{[z] + [C(\zeta)] \cdot [u]\}\}_i \quad \textit{where}$$
$$[u] := ([v_1], \cdots, [v_{i-1}], [y_i], \cdots, [y_n])^\top.$$

If $[v] \overset{\circ}{\subset} [y]$, *then* R *and every matrix* $\mathcal{A}(\zeta) \in [\mathcal{A}(\zeta)]$, $\zeta \in [\zeta]$ *is regular, so every matrix* $A(p)$, $p \in [p]$ *is regular, and for every* $p \in [p]$ *the unique solution* $\hat{x} = A^{-1}(p)b(p)$ *of* $A(p) \cdot x = b(p)$ *satisfies* $\hat{x} \in \tilde{x} + [v]$.

Now, we give an algorithm for computing an outer solution for the system (1)

Algorithm 1. Parametric interval linear systems

1. Initialization
   $\check{b}$:=mid(reduce($[b(\zeta)]$)); $\check{A}$:=mid(reduce($[A(\zeta)]$))
2. Computation of an approximate mid-point solution
   $\tilde{x} = R \cdot \check{b}$; ($R \approx \check{A}^{-1}$)
3. Computation of an enclosure $[C] \in I\mathbb{R}^{n\times n}$
   $[C] := I - R \cdot [\mathcal{A}^{(0)}] - \sum_{\nu=1}^{m} (R \cdot [\mathcal{A}^{(\nu)}])[\zeta_\nu]$
4. Computation of an enclosure $[z]$
   $[z]$:=$R\cdot([\ell^{(0)}]-[\mathcal{A}^{(0)}]\cdot\tilde{x})+\sum_{\nu=1}^{m} R\cdot([\ell^{(\nu)}]-[\mathcal{A}^{(\nu)}]\cdot\tilde{x})[\zeta_\nu]$
5. Verification step
   $[v] := [z]$
   max= 1
   repeat
   $[y] := [v]$
   for $i = 1$ **to** n **do**
   $[v_i] = [z_i] + [C(Row(i))] \cdot [v]$
   max++
   until $[v] \overset{\circ}{\subset} [y]$ *or max*≥ 10
6.
   if $([v] \overset{\circ}{\subset} [y])$ **then**
   $\hat{x} \in \tilde{x} + [v]$
   else *no inclusion can be computed*

5. Numerical Examples

Example 3.

$$\begin{pmatrix} -(p_1+1)p_2 & p_1p_3 & p_2 \\ p_2p_4 & p_2^2 & 1 \\ p_1p_2 & p_3p_5 & \sqrt{p_2} \end{pmatrix} \cdot x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$[p] = ([1, 1.2], [2, 2.2], [0.5, 0.51], [0.39, 0.40], [0.39, 0.40])^T \in I\mathbb{R}^5$

New Method	*Kolev's Method [10]*
$[\underline{0.055}479, \underline{0.066}083]$	$[\underline{0.055}081, \underline{0.066}443]$
$[\underline{0.07}6096, \underline{0.090}512]$	$[\underline{0.07}5930, \underline{0.090}906]$
$[\underline{0.55}7139, \underline{0.60}6451]$	$[\underline{0.55}5988, \underline{0.60}7462]$

Example 4.

$$\begin{pmatrix} -(p_1+p_2)p_4 & p_2p_4 \\ p_5 & p_3p_5 \end{pmatrix} \cdot x = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$[p] = ([0.96, 0.98], [1.92, 1.96], [0.96, 0.98], [0.48, 0.5], [0.48, 0.5])^T \in I\mathbb{R}^5$

New Method	*Kolev's Method*
$[\underline{0.3}74648, \underline{0.4}56641]$	$[\underline{0.3}67181, \underline{0.4}64108]$
$[\underline{1.6}21478, \underline{1.7}29391]$	$[\underline{1.6}13711, \underline{1.7}37157]$

Example 5.

$$\begin{pmatrix} -(p_1+1)p_2 & p_1p_3 & \exp(p_2) \\ p_2p_4 & p_2^2 & 1 \\ p_1p_2 & p_3p_5 & \sqrt{p_2} \end{pmatrix} \cdot x = \begin{pmatrix} \cos(p_1) \\ 1 \\ 1 \end{pmatrix},$$

$[p] = ([1, 1.2], [2, 2.2], [0.5, 0.51], [0.39, 0.40], [0.39, 0.40])^T \in I\mathbb{R}^5$

New Method	*Kolev's Method*
$[\underline{0.26}1268, \underline{0.32}57415]$	$[\underline{0.26}0297, \underline{0.32}6198]$
$[\underline{0.10}3746, \underline{0.14}6084]$	$[\underline{0.10}2870, \underline{0.14}7174]$
$[\underline{0.16}9010, \underline{0.24}1066]$	$[\underline{0.16}6773, \underline{0.24}4037]$

6. Conclusions

The problem of solving parametric linear systems of equations whose elements are nonlinear function of interval parameters is very important in practical applications. Well-known classical methods, such as interval version of Gauss elimination, fail since they compute enclosure for the solution set (4) which is generally much larger than solution set (3). A simple method for determining an outer solution to the linear system considered has been suggested in section (4). An algorithm is presented and some examples are solved and compared with Kolev's method, from these examples we saw that our method gives (may be not at all) better enclosures than the method by Kolev [10]. Our method can be applied to big real life problems such as structural engineering [2] without any problems.

References

[1] Alefeld, G.; Herzberger, J.: *Introduction to Interval Computations*. Academic Press, 1983.

[2] Corliss, G.; Foley, C.; R. B. Kearfott: *Formulation for Reliable Analysis of Structural Frames*. Reliable Computing, vol. 13, no. 2, pp.125-145, 2007.

[3] Dessombz O., et al.: *Analysis of mechanical systems using interval computations applied to finite element methods*, Journal of Sound and Vibration 239 (2001) 5, 949-968.

[4] Hansen, E. R. : *Generalized Interval Arithmetic*, in Nickel, K. L. (ed.), Interval Mathematics, Vol. 29 of lecture notes in computer science, page 7-18, Springer-Verlag, Berlin, 1975.

[5] Hansen, E. R. : *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc. 1992.

[6] El-Owny, H.: *Hansen's Generalized Interval Arithmetic Realized in C-XSC* , Preprint 2006/2, Universität Wuppertal, 2006.
(http://www.math.uni-wuppertal.de/wrswt/preprints/prep_06_2.pdf)

[7] Hofschuster, W.; Krämer, W.; Wedner, S.; Wiethoff, A.: *C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing*. Preprint 2001/1,

Wissenschaftliches Rechnen / Softwaretechnologie, Universität Wuppertal, 2001.
(http://www.math.uni-wuppertal.de/wrswt/preprints/prep_01_1.pdf)

[8] Hofschuster, W.; Krämer, W., *C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing.* In: Numerical Software with Result Verification, R. Alt, A. Frommer, B. Kearfott, W. Luther (eds), Springer Lecture Notes in Computer Science 2991, pp. 15-35, 2004.

[9] Kolev, L.: *A Method for Outer Interval Solution of Linear Parametric Systems.* Reliable Computing, vol. 10, nr. 3, pp. 227-239, 2004.

[10] Kolev, L.: *Solving Linear Systems Whose Elements are Nonlinear Functions of Intervals.* Numerical Algorithms 10, nr. 1-4, pp. 199-212, 2004.

[11] Krämer, W.; Popova, E. D.: *Zur Berechnung von verlässlichen Außen- und Inneneinschließungen bei parameterabhängigen linearen Gleichungssystemen.* PAMM - Proceedings in Applied Mathematics and Mechanics, vol. 4, issue 1, pp. 670-671, 2004.

[12] Krämer, W.: *Generalized Intervals and the Dependency Problem.* PAMM - Proceedings in Applied Mathematics and Mechanics, vol. 6, pp. 683-684, 2006.

[13] Moore, R.: *Interval analysis*. Prentice-Hall, Inc. Englewood Cliffs, N. J. , 1966.

[14] Neumaier, A.; Pownuk, A.: *Linear Systems with Large Uncertainties with Applications to Truss Structures.* Reliable Computing, vol. 13, issue 2, pp. 149-171, 2007.

[15] Popova, E.; Datcheva, M.; Iankov, R.; Schanz T.: *Mechanical Models with Interval Parameters.* In K. Gürlebeck L. Hempel C. Könke (Eds.) IKM2003: Digital Proceedings of 16th International Conference on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering, ISSN 1611-4086, Weimar, 2003.
(http://euklid.bauing.uni-weimar.de/papers/36/M36.pdf)

[16] Popova, E.: *Improved Parametric Fixed-Point Iteration,* Preprint Inst. of Mathematics & Informatics, BAS, Sofia, 2003.

[17] Popova, E.; Krämer, W.: *Parametric Fixed-Point Iteration Implemented in C-XSC.* Preprint 2003/3, Universität Wuppertal, 2003.

[18] Popova, E.: *Generalizing the Parametric Fixed-Point Iteration.* Proceeding in Applied Mathematics and Mechanics (PAMM) 4, issue 1, pp. 680-681, 2004.

[19] Popova, E.; Krämer, W.: *Inner and Outer Bounds for the Solution Set of Parametric Linear Systems.* Journal of Computational and Applied Mathematics, vol. 199, issue 2, pp. 310-316, 2007.

[20] Rump, S.: *Verification methods for dense and sparse systems of equations.* In: Topic in Validated Computations, Herzberger, J. (ed.), Oldenburg, North-Holland, 1994.

Guaranteed Bounds for Uncertain Systems: Methods Using Linear Lyapunov-like Functions, Differential Inequalities and a Midpoint Method

Marc Gennat and Bernd Tibken
Faculty of Electrical, Information and Media Engineering
University of Wuppertal, D-42097 Wuppertal, Germany
{gennat,tibken}@uni-wuppertal.de

Abstract

In general, models of biological or technical applications are represented by nonlinear systems. Moreover, these systems contain multiple uncertain or unknown parameters. These uncertainties are the reason for some numerical and analytical problems in finding guaranteed bounds for the solution of the state space representation. Unfortunately, several industrial applications are demanding exactly these guaranteed bounds in order to fulfil regulations set by the state authorities. To get an idea of the solution of systems with uncertainties the numerical integration of the system's differential equations has to be done with randomly selected values for the unknown parameters. This computation is done several times, in some circumstances more than a thousand times. This approach is well known as the Monte-Carlo method, but this stochastic approach cannot deliver guaranteed bounds for the domain of the system's solution. Thus, we developed a method to find guaranteed bounds which uses linear Lyapunov-like functions to solve this problem. In this work we combine this method with a theory first introduced by Müller. Differential inequalities are used by Müller to obtain guaranteed bounds. Intersecting the results of both methods provides improved and tight bounds for the original uncertain system. Another approach is shown using a midpoint method providing guaranteed bounds. We achieve guaranteed and finite simulation bounds as a result of our approaches. The results can be used as an initial interval for further methods based on interval arithmetic. An example of a bioreactor with two state variables is shown in this paper to illustrate the methods.

1 Introduction

The simulation of nonlinear systems with uncertain or unknown parameters is demanded by many technical, biological and chemical applications. This is the main reason why we have developed a novel method to derive guaranteed bounds for such systems. The simulation for such systems cannot be done in one calculation cycle as it is done in the case of known parameters. The time dependent set of all simulated state variables with all possible parameter combinations is required. We consider a system of nonlinear ordinary differential equations which represents an uncertain dynamical system in state space representation

$$\dot{x}(t) \quad = \quad f(x(t);p) \text{ with } x(0) = x_0, \tag{1}$$

where the vector $x \in \mathbb{R}^n$ and function $f(x(t);p)$ represent the time dependent state vector and the nonlinearity, respectively. The vector of uncertain or unknown parameters is given by $p \in \mathbb{R}^q$, which is contained in an interval vector, in other words the uncertain parameters can vary between a lower bound $\underline{p}$ and an upper bound $\overline{p}$ with $p \in [\underline{p}, \overline{p}]$.

2 Simulation Method Using Müller's Theorem

In order to find guaranteed bounds we apply results from the theory of differential inequalities, which was first presented by Müller [3, 4]. The aim is to find lower and upper functions to bound the right hand side of the original system, which is afflicted with uncertain or unknown parameters. In order to bound $\dot{x}(t) = f(x(t);p)$ the right hand side of the given system must be enclosed componentwise by lower and upper bound functions

$$u_i(t) \ \leq \ x_i(t) \leq w_i(t) \ \forall \ 0 < t < \delta \text{ with } \delta > 0, \tag{2}$$

where $u_i(t)$ represents the lower bound function and $w_i(t)$ the upper bound function for the i-th state space function. To find these functions the differential inequalities with $\dot{u}(t)$ and $\dot{w}(t)$ have to fulfil componentwise the conditions

$$\dot{u}_i(t) \leq f_i(z;p) \text{ if } u(t) \leq z \leq w(t) \text{ and } u_i(t) = z_i, \tag{3}$$
$$\dot{w}_i(t) \geq f_i(z;p) \text{ if } u(t) \leq z \leq w(t) \text{ and } w_i(t) = z_i \tag{4}$$

with $i = 1, \cdots, n$. To do that, the right hand sides $f_i(z;p)$ must be modified to match the requirements. This is done by replacing the state variables z_ν by the state variables for the lower and upper bound functions u_ν or w_ν with $\nu = 1, \cdots, n, \nu \neq i$, to minimize the right hand side for $\dot{u}_i$ and to maximize for $\dot{w}_i$. If ν is equal to i, z_i must be set to u_i for the differential equation of $\dot{u}_i$ respectively it must be set to w_i for the right hand side of $\dot{w}_i$. Moreover, the uncertain or unknown parameter intervals p_i must be replaced by the infimum of the parameter range $\underline{p}_i$ or the supremum $\overline{p}_i$ with $i = 1, \cdots, q$. The replacement of the parameters and state variables for the right hand side of the differential inequalities (3) and (4) must satisfy the minimization

$$\begin{aligned} \dot{u}_i(t) \quad = \quad & \min \quad f_i(z;p) \\ & \text{s.t. } z_i = u_i(t) \\ & u(t) \leq z \leq w(t) \\ & \underline{p} \leq p \leq \overline{p} \end{aligned} \tag{5}$$

and the maximization

$$\begin{aligned} \dot{w}_i(t) \quad = \quad & \max \quad f_i(z;p). \\ & \text{s.t. } z_i = w_i(t) \\ & u(t) \leq z \leq w(t) \\ & \underline{p} \leq p \leq \overline{p} \end{aligned} \tag{6}$$

This leads to right hand sides of $\dot{u}_i$ and $\dot{w}_i$, where the state variables z are replaced by u or w to minimize (5) and maximize (6). Only the i-th state variable in the i-th lower bound function must be set to u_i and the i-th state variable in the i-th upper bound function to w_i. At last the uncertain parameters must be chosen in a way to minimize or maximize the lower and upper bound functions. With these optimizations performed componentwise for each state space function of the original system (1) we achieve a system of differential equations, which has twice as many equations as the original system. To compute the lower and upper bounds for the uncertain system, the new system

$$\begin{pmatrix} \dot{u}_1(t) \\ \vdots \\ \dot{u}_n(t) \\ \dot{w}_1(t) \\ \vdots \\ \dot{w}_n(t) \end{pmatrix} = \begin{pmatrix} \min f_1(z;p) \\ \vdots \\ \min f_n(z;p) \\ \max f_1(z;p) \\ \vdots \\ \max f_n(z;p) \end{pmatrix} \tag{7}$$

of the order of $2\,n$ has to be solved. Thus, we obtain as a result of the simulation the lower bounds $u(t)$ and the upper bounds $w(t)$ for each component of the uncertain system. Unfortunately, this method may lead to unstable upper bounds for $t \to \infty$ in general, but for small time instances the method by Müller provides very good results in the form of guaranteed and tight bounds for the given system (1).

3 Simulation Method Using a Midpoint Method for Differential Inequalities

The state space representation (1) for the system class under consideration is assumed to have the form

$$\dot{x}(t) = f(x(t);p) = \begin{pmatrix} f_1(x_1(t), \cdots, x_n(t); p_1, \cdots, p_q) \\ \vdots \\ f_m(x_1(t), \cdots, x_n(t); p_1, \cdots, p_q) \end{pmatrix}. \tag{8}$$

We have to find upper and lower bounds for the given system. These bounds should provide a guaranteed enclosure of all possible solutions with all variations of the uncertain parameters. We assume f has continuous derivatives of at least one order. Thus, we define the differential equation with the midpoint of the uncertain parameters $\breve{p} = \text{mid}(p)$ as

$$\dot{\breve{x}}(t) \quad = \quad f(\breve{x}(t);\breve{p}) \text{ with } \breve{x}(0) = x(0) = x^0. \tag{9}$$

To bound the right hand side of (1) we first have to reformulate the problem using the new state variable $z(t) = x(t) - \breve{x}(t)$, which represents the deviation from the state variable x. Thus, the system can be defined as

$$\dot{z}(t) \quad = \quad \dot{x}(t) - \dot{\breve{x}}(t) = f(x(t);p) - f(\breve{x}(t);\breve{p}). \tag{10}$$

At this point we use an advanced interval method which comes from bounding the remainder in a Taylor series [3, 7, 8, 13]. In this particular case we rewrite $\dot{z}(t)$ as

$$\begin{aligned} \dot{z}(t) = & \; f(\breve{x}(t);\breve{p}) + \left(\frac{\partial f}{\partial x, p}(\xi_{x(t)};\xi_p)\right)^T \begin{pmatrix} x - \breve{x} \\ p - \breve{p} \end{pmatrix} - f(\breve{x}(t);\breve{p}) \\ = & \; \frac{\partial f}{\partial x}(\xi_{x(t)};\xi_p)(x - \breve{x}) + \frac{\partial f}{\partial p}(\xi_{x(t)};\xi_p)(p - \breve{p}) \\ = & \; \frac{\partial f}{\partial x}(\xi_{x(t)};\xi_p)(z) + \frac{\partial f}{\partial p}(\xi_{x(t)};\xi_p)(p - \breve{p}) \end{aligned} \tag{11}$$

where ξ_x lies between x and $\breve{x}$ and ξ_p lies between p and $\breve{p}$. The given system can then be rewritten as a linear time variant system

$$\begin{aligned} \dot{z}(t) \quad &= \quad A(t) \cdot z + b(t) \text{ with} \\ A(t) \quad &= \quad \frac{\partial f}{\partial x}(\xi_{x(t)};\xi_p) \text{ and} \\ b(t) \quad &= \quad \frac{\partial f}{\partial p}(\xi_{x(t)};\xi_p)(p - \breve{p}), \end{aligned}$$

where $A(t)$ is an interval matrix and $b(t)$ an interval vector. For bounding the solution for all possible parameters we have to initialize ξ_x and ξ_p as intervals. To make sure that ξ_x and ξ_p cover all possible points, we define ξ_x and ξ_p as intervals with

$$\begin{aligned} \xi_{x(t)} \quad &= \quad [\underline{z}(t) - \breve{x}(t)\,;\, \overline{z}(t) - \breve{x}(t)] \;=\; [\underline{x}(t)\,;\, \overline{x}(t)] \text{ and} \\ \xi_p \quad &= \quad [\underline{p}\,;\, \overline{p}]. \end{aligned}$$

The right hand side of (11) provides an interval solution which leads to a guaranteed enclosure of the solution of the right hand side of the original problem (1) by adding z and $\check{x}$, the solution of the midpoint differential equation (9). In general this interval set is afflicted with overestimation. Thus, we have to rewrite the differential equation (11) as

$$\begin{aligned}\dot{v}(t) &= \inf(\dot{x}(t) - \dot{\check{x}}(t)) \quad (12)\\ &= \inf\left(\frac{\partial f}{\partial x}(\xi_{x(t)};\xi_p)v(t) + \frac{\partial f}{\partial p}(\xi_{x(t)};\xi_p)(p-\check{p})\right),\\ \dot{w}(t) &= \sup(\dot{x}(t) - \dot{\check{x}}(t)) \quad (13)\\ &= \sup\left(\frac{\partial f}{\partial x}(\xi_{x(t)};\xi_p)w(t) + \frac{\partial f}{\partial p}(\xi_{x(t)};\xi_p)(p-\check{p})\right),\end{aligned}$$

where $\inf$ returns the infimum and $\sup$ the supremum value of an interval. Now we can define an extended system of differential equations, which provides the guaranteed bounds directly. The extended system is given as

$$\begin{aligned}\dot{\check{x}}(t) &= f(\check{x}(t);\check{p}) \quad (14)\\ \dot{v}(t) &= \inf\left(\frac{\partial f}{\partial x}([\check{x}+v(t),\check{x}+w(t)];p)[v(t),w(t)]\right.\\ &\quad \left. + \frac{\partial f}{\partial p}([\check{x}+v(t),\check{x}+w(t)];p)(p-\check{p})\right) \quad (15)\\ \dot{w}(t) &= \sup\left(\frac{\partial f}{\partial x}([\check{x}+v(t),\check{x}+w(t)];p)[v(t),w(t)]\right.\\ &\quad \left. + \frac{\partial f}{\partial p}([\check{x}+v(t),\check{x}+w(t)];p)(p-\check{p})\right) \quad (16)\end{aligned}$$

with $\check{x}(0) = x(0) = x^0$, $v(0) = w(0) = 0$ and in the i-th equation of $\dot{v}(t)$ the interval $[v_i(t), w_i(t)]$ has to replaced by v_i respectively w_i in the in the i-th equation of $\dot{w}(t)$ [3]. Reducing overestimation for the computation of $\dot{v}(t)$ and $\dot{w}(t)$ is a difficult job, which can be done by an appropriate factorisation of the right hand sides of the extended system. Primary objective is to reduce the number of evaluations of interval variables. If each interval variable is evaluated only one time, one will obtain the best bounds for the given system with minimal overestimation. This objective cannot be fulfilled for all classes of systems, nevertheless one can use the technique of bisection for reducing overestimation [8, 11, 13].

4 Upper Bounds by Linear Lyapunov-like Functions

The state space representation (1) for the system class under consideration is assumed to have the form

$$\dot{x}(t) = f(x(t);p) = A\,x(t) + b\,g(x(t);p) + d \quad (17)$$

where $A \in \mathbb{R}^{n\times n}$ is the system matrix of the linear part, the function $g(x(t);p), \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ represents the nonlinear part, $b \in \mathbb{R}^n$ is a constant vector and the vector d contains all constant parts that do not depend on the state variables, inputs or parameters. We assume further that the system given by (17) leaves the positive orthant of $\mathbb{R}^n$ invariant. Thus, we have $x(t) \geq 0 \quad \forall\, t$ if $x(0) \geq 0$ is fulfilled. Our goal is to find an easy way to compute upper bounds for the state variables. This is a difficult problem, which can in principle be solved through the Monte-Carlo Method [1, 2] with the drawback that the bounds are not guaranteed. A recent method is interval arithmetic [7, 8, 11, 12, 13], which also may lead to unstable upper bounds.

4.1 Finding an Auxiliary Function

In order to overcome these problems we define a linear Lyapunov-like function

$$v(t) = c^T\,x(t), \quad (18)$$

with $c^T = (c_1, \cdots, c_n)$, $c_i \geq 0$ with $i = 1, \cdots, n$ and $\|\,c\,\| > 0$. Now we compute the time derivative of (18) along the trajectories of (17). This results in

$$\begin{aligned}\dot{v}(t) = c^T\,\dot{x}(t) &= c^T(A\,x(t) + b\,g(x(t);p) + d)\\ &= c^T\,A\,x(t) + c^T\,b\,g(x(t);p) + c^T\,d. \quad (19)\end{aligned}$$

If we now assume

$$\begin{aligned}c^T\,b &= 0 \text{ and} \quad (20)\\ \text{all components } (c^TA)_i &< 0 \text{ with } i = 1, \cdots, n, \quad (21)\end{aligned}$$

we compute

$$\dot{v}(t) = (c^TA)\,x(t) + c^Td, \quad (22)$$

thus the nonlinear function g is eliminated, because the condition (20) demands $c^Tb = 0$. Due to the fact that $x(t) > 0\ \forall\, t$ and the assumption (21), we can find an upper bound for the right hand side of (22). This results in

$$\begin{aligned}\dot{v}(t) &= c^T\,d - \left[\frac{-c^T\,A\,x(t)}{c^T\,x(t)}\right]\cdot v(t)\\ \dot{v}(t) &= c^T\,d - \gamma\,v(t) \quad (23)\\ \text{with } \gamma &= \frac{-c^T\,A\,x(t)}{c^T\,x(t)}.\end{aligned}$$

Our aim is to bound $\dot{v}(t)$ upwards, thus we have to maximize the right hand side of (23) which leads to minimizing γ, according to

$$\begin{aligned}\gamma &= \min_{x>0\,,\ x\neq 0}\left(-\frac{c^T\,A\,x}{c^T\,x}\right), \text{ which is equivalent to}\\ \gamma &= \min_{x>0, c^Tx=1}\left(-c^T\,A\,x\right). \quad (24)\end{aligned}$$

The linear optimization problem (24) has the solution

$$\gamma = \min\left(\frac{-(c^T A)_1}{c_1}, \frac{-(c^T A)_2}{c_2}, \cdots, \frac{-(c^T A)_n}{c_n}\right). \quad (25)$$

Now we are in the position to find an upper bound for the linear Lyapunov-like function itself. Therefore we have to compute a solution for the differential inequality

$$\dot{v}(t) \leq c^T\, d - \gamma \cdot v(t). \quad (26)$$

Using the Gronwall Lemma [6] we calculate an upper bound as

$$v(t) \leq v(0)e^{-\gamma t} + c^T d \int_0^t e^{-\gamma(t-\tau)} d\tau \quad (27)$$

which results in

$$v(t) \leq v(0)e^{-\gamma t} + \frac{1}{\gamma} c^T d\left(1 - e^{-\gamma t}\right) = \tilde{v}(t)\,. \quad (28)$$

Thus, $\tilde{v}(t)$ is an upper bound for the linear Lyapunov-like function $v(t)$. As a result of the guaranteed bound (28) the state variables bounds are given by

$$0 \;\leq\; x_i \;\leq\; \frac{\tilde{v}_i(t)}{c_i} \quad \forall \quad i = 1, \cdots, n\,. \quad (29)$$

This equation represents the guaranteed bounds for the state variables of the given nonlinear system.

4.2 Optimizing the Bounds

The essential part of the problem is solved. Guaranteed bounds for nonlinear systems with unknown or uncertain parameters are found. But we can do more. We use c to optimize the bounds for time to infinity ($t \to \infty$). According to (29) we find guaranteed bounds for the state variables as $0 \leq x_i(t) \leq \frac{\tilde{v}_i(t)}{c_i}$. So we can formulate the minimization of each state variable x_i for an infinite time interval as an optimization problem

$$\begin{aligned} \min_c \quad & \left(\lim_{t\to\infty}\left(\frac{\tilde{v}_i(t)}{c_i}\right)\right) \text{ subject to} \quad (30)\\ c^T b \;&=\; 0, \quad c > 0,\\ (c^T A)_i \;&<\; 0 \text{ for } i = 1, \cdots, n \text{ and}\\ \gamma \;&=\; \min_i\left(\frac{-(c^T A)_i}{c_i}\right). \quad (31) \end{aligned}$$

The constrained optimization problem (30, 31) leads to a vector c for optimal bounds of the state variables for $t \to \infty$. Moreover, $\tilde{v}_i(t)$ can be simplified as

$$\lim_{t\to\infty} \tilde{v}_i(t) = \lim_{t\to\infty}\left(v(0)\underbrace{e^{-\gamma t}}_{\to 0} + \frac{c^T d}{\gamma} - \frac{c^T d}{\gamma}\underbrace{e^{-\gamma t}}_{\to 0}\right) = \frac{c^T d}{\gamma}.$$

Using this extension the optimization problem is rewritten as

$$\begin{aligned} \min_c \quad & \left(\frac{c^T d}{\gamma\, c_i}\right) \text{ subject to} \quad (32)\\ c^T b \;&=\; 0, \quad c > 0,\\ (c^T A)_i \;&<\; 0 \text{ for } i = 1, \cdots, n \text{ and}\\ \gamma \;&=\; \min_i\left(\frac{-(c^T A)_i}{c_i}\right). \end{aligned}$$

The last constraint, $\gamma = \min_i\left(\frac{-(c^T A)_i}{c_i}\right)$, is non-differentiable and we rewrite the problem as an extended optimization problem. This results in

$$\begin{aligned} \min_{c,\hat{\gamma}} \quad & \left(\frac{c^T d}{\hat{\gamma}\, c_i}\right) \text{ subject to} \quad (33)\\ c^T b \;&=\; 0, c > 0,\\ \text{all components of } (c^T A)_i \;&<\; 0 \text{ for } i = 1, \cdots, n \text{ and}\\ \hat{\gamma} \;&\leq\left(\frac{-(c^T A)_i}{c_i}\right) \text{for } i = 1, \cdots, n. \end{aligned}$$

For each state variable x_i a single constrained optimization has to be done, which provides optimal c and $\hat{\gamma}$ for the specified state variable. The simulation results using c and $\hat{\gamma}$ lead to optimal bounds for this state variable. Moreover, these bounds can be used as a starting point for other methods, e.g. interval arithmetic. These methods profit a lot from optimal bounds, because overestimation is a serious problem in simulations of nonlinear systems with uncertain parameters.

5 Example

The term bioreactor [18, 19] refers to a system that supports a biologically active environment. A bioreactor is a vessel in which a chemical process is carried out, which involves micro-organisms or bio-chemically active substances derived from such micro-organisms. This process can either be aerobic or anaerobic. The given example is a system of two state variables, which represent the concentration of the bio-chemically active substances, such as bacteria, and the concentration of nutrient substrate. The system is given by

$$\dot{S}(t) = D\,(S_r - S(t)) - \frac{\mu}{Y} X(t), S(0) = S^0, \quad (34)$$

$$\dot{X}(t) = (\mu - D)X(t), X(0) = X^0 \text{ with} \quad (35)$$

$$\mu = \mu_m \frac{S(t)}{K_0 + S(t) + K_1 S^2(t)} \text{ and } x = \begin{pmatrix} S(t) \\ X(t) \end{pmatrix}$$

as the state space vector. The initial conditions for the substrate concentration are given by $S^0 = 1$ and the concentration of the bio-chemically active substances, in other words

the bacteria concentration is given by $X^0 = 1$. The other parameters are given by $\mu_m = 0.2$, $Y = 0.5$ and $S_R = 100$. The uncertain parameters are the saturation coefficient K_0 and K_1, which can vary in the intervals $K_0 = [0.16, 0.24]$ and $K_1 = [0.008, 0.012]$, thus the midpoints of these intervals are $\check{K}_0 = 0.2$ and $\check{K}_1 = 0.01$.

5.1 Guaranteed Bounds corresponding to Müller's Theorem

The guaranteed bounds for the given example corresponding to Müller's Theorem can be calculated by the simulation of the upper and lower bound functions for the differential equations (5) and (6). To calculate the required minimization and maximization we rewrite the original state space representation as

$$\dot{z}_1 = D\,(S_r - z_1(t)) - \frac{\mu_m}{Y}\,\frac{z_1(t)z_2(t)}{K_0 + z_1(t) + K_1 z_1^2(t)} \text{ and}$$

$$\dot{z}_2 = \left(\mu_m \frac{z_1(t)}{K_0 + z_1(t) + K_1 z_1^2(t)} - D\right) z_2(t)$$

with $z_1 = S(t)$, $z_2 = X(t)$, $z_1(0) = S^0$ and $z_2(0) = X^0$. Referring to the extended system of differential equations (7) the minimization and maximization of (5) and (6) leads to

$$\dot{u}_1 = D\,(S_r - u_1(t)) - \frac{\mu_m}{Y}\,\frac{u_1(t)w_2(t)}{K_{0_{min}} + u_1(t) + K_{1_{min}} u_1^2(t)}$$

$$\dot{u}_2 = \mu_m \frac{u_1(t)u_2(t)}{K_{0_{max}} + w_1(t) + K_{1_{max}} w_1^2(t)} - D\,u_2(t)$$

$$\dot{w}_1 = D\,(S_r - w_1(t)) - \frac{\mu_m}{Y}\,\frac{w_1(t)u_2(t)}{K_{0_{max}} + w_1(t) + K_{1_{max}} w_1^2(t)}$$

$$\dot{w}_2 = \mu_m \frac{u_1(t)w_2(t)}{K_{0_{min}} + u_1(t) + K_{1_{min}} u_1^2(t)} - D\,w_2(t),$$

where the uncertain parameters K_0 and K_1 have to be replaced by the infimum and supremum of the parameter intervals. This replacement has to be done to fulfil the requirements of the minimization and maximization of (5) and (6). Using this expanded system, upper and lower bounds for the original system can be computed.

5.2 Guaranteed Bounds Using a Midpoint Method for Differential Inequalities

For this method we applied the heterotrophic yield Y as the uncertain parameter, which can vary in the interval $Y = [0.45, 0.55]$, thus the midpoint of this interval is $\check{Y} = 0.5$. To apply our method we have to compute the derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial p}$. The derivatives are

$$\frac{\partial f_1}{\partial S} = -D + \frac{\mu_m X_{var}}{\left(\frac{K_0}{S_{var}} + 1 + K_1 \cdot S_{var}\right)^2 Y_{var}} \left(\frac{-K_0}{S_{var}^2} + K_1\right)$$

$$\frac{\partial f_1}{\partial X} = \frac{-\mu_m}{\frac{K_0}{S_{var}} + 1 + K_1 \cdot S_{var}} \frac{1}{Y_{var}}$$

$$\frac{\partial f_1}{\partial Y} = \frac{\mu_m}{\frac{K_0}{S_{var}} + 1 + K_1 \cdot S_{var}} \frac{X_{var}}{Y_{var}^2}$$

$$\frac{\partial f_2}{\partial S} = \frac{\mu_m}{(\frac{K_0}{S_{var}} + 1 + K_1 \cdot S_{var})^2} X_{var} \left(\frac{-K_0}{S_{var}^2} + K_1\right)$$

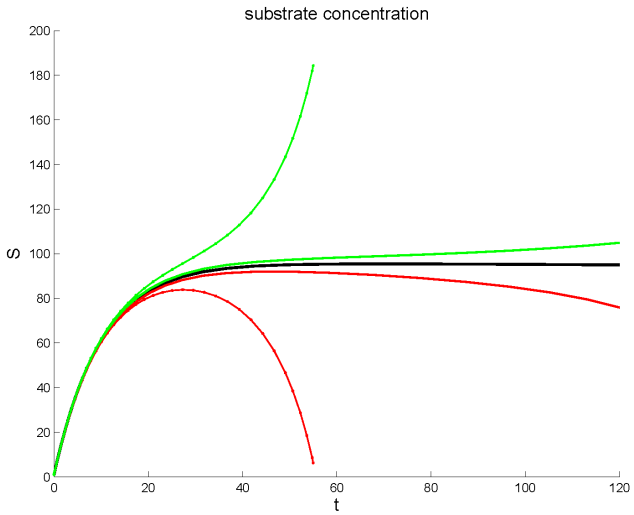


Figure 1. Simulations of the substrate concentration using the Midpoint Method with bisection

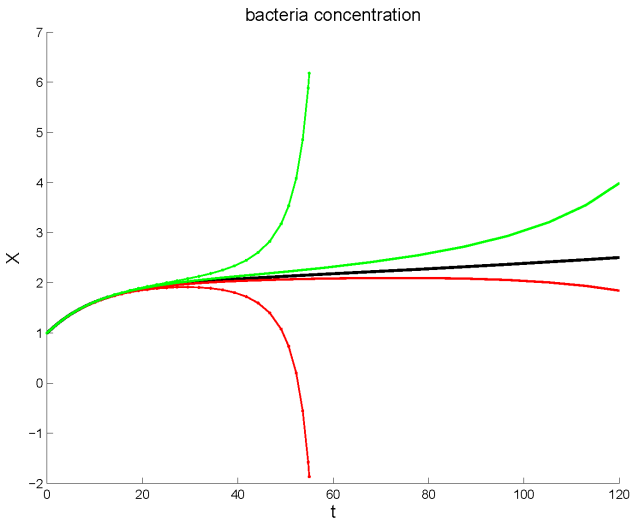


Figure 2. Simulations of the bacteria concentration using the Midpoint Method with bisection

$$\frac{\partial f_2}{\partial X} = \left(\frac{-\mu_m}{\frac{K_0}{S_{var}} + 1 + K_1 \cdot S_{var}} - D\right) X_{var}$$
$$\frac{\partial f_2}{\partial Y} = 0.$$

The abbreviations $z_\nu = [\check{x}_\nu + v_\nu, \check{x}_\nu + w_\nu]$ for $\nu = 1, 2$ will be used in the sequel. With the derivatives, the extended system has the form

$$\dot{\check{x}}_1 = D\,(S_r - S) - \frac{\mu}{Y} X \tag{36}$$
$$\dot{\check{x}}_2 = (\mu - D) X \tag{37}$$
$$\dot{v}_1 = \inf\left(\frac{\partial f_1}{\partial S}(z) v_1 + \frac{\partial f_1}{\partial X}(z)[v_2, w_2] + \frac{\partial f_1}{\partial Y}(Y - \check{Y})\right) \tag{38}$$
$$\dot{v}_2 = \inf\left(\frac{\partial f_2}{\partial S}(z)[v_1, w_1] + \frac{\partial f_2}{\partial X}(z) v_2\right) \tag{39}$$
$$\dot{w}_1 = \sup\left(\frac{\partial f_1}{\partial S}(z) w_1 + \frac{\partial f_1}{\partial X}(z)[v_2, w_2] + \frac{\partial f_1}{\partial Y}(Y - \check{Y})\right) \tag{40}$$
$$\dot{w}_2 = \sup\left(\frac{\partial f_2}{\partial S}(z)[v_1, w_1] + \frac{\partial f_2}{\partial X}(z) w_2\right). \tag{41}$$

In Fig. 1 and Fig. 2 the black line represents the simulation result using $\check{Y}$ as the midpoint of the uncertain parameter. The dotted lines represent the upper and lower bounds. The solution of the right hand side of this system leads to much overestimation, thus the direct implementations of (36-41) diverge just before the 60^{th} time step. With the appropriate factorisation of the right hand sides the overestimation can be reduced significantly. The simulation of the factorised extended system is represented by the undotted lines and provides guaranteed bounds with much less overestimation.

5.3 Guaranteed Bounds provided by Linear Lyapunov-like functions

Taking the matrices and vectors following from the model (34,35) the notation leads to

$$x(t) = \begin{pmatrix} S(t) \\ X(t) \end{pmatrix} \qquad A = \begin{pmatrix} -D & 0 \\ 0 & -D \end{pmatrix}$$
$$b = \begin{pmatrix} -\frac{1}{Y} \\ 1 \end{pmatrix} \qquad d = \begin{pmatrix} D\ S_R \\ 0 \end{pmatrix}$$

and the nonlinear function is defined as

$$g(t) = \mu_m \frac{S(t)}{K_0 + S(t) + K_1 S^2(t)} X(t). \tag{42}$$

We are applying the constrained optimization

$$\min_{c,\gamma} \left(\frac{c^T d}{\gamma\, c_i}\right) \text{ subject to} \tag{43}$$
$$c^T b = 0, \quad c > 0,$$
$$(c^T A)_i < 0 \text{ for } i = 1, 2 \text{ and}$$
$$\gamma \leq \left(\frac{-(c^T A)_i}{c_i}\right) \text{ for } i = 1, 2.$$

The constrained optimization leads to a vector c with a corresponding optimal γ, which delivers optimal bounds for the state variable simulation for $t \to \infty$. In this example the optimization leads to $c = (Y \quad 1)^T$ and $\gamma = 0.1$. In Section 4 we proved the existence of a bounded $\tilde{v}(t)$, which can be used to determine the range of the state variables. According to (28) $\tilde{v}(t)$ is derived as

$$\tilde{v}(t) = v(0) e^{-\gamma t} + \frac{1}{\gamma} Y c^T d \left(1 - e^{-\gamma t}\right)$$
$$\Rightarrow \tilde{v}(t) = (Y + 1) e^{-0.1t} + 10\, Y\, D\, S_R \left(1 - e^{-0.1t}\right).$$

Using $\tilde{v}(t)$ to determine the upper bound of the substrate concentration, we get

$$0 \leq S(t) \leq \frac{\tilde{v}(t)}{c_1} \Rightarrow$$
$$0 \leq S(t) \leq \left(1 + \frac{1}{Y} - 10 D S_R\right) e^{-0.1t} + 10 D S_R. \tag{44}$$

Bounds for bacteria concentration result in

$$0 \leq X(t) \leq \frac{\tilde{v}_a(t)}{c_2} \Rightarrow$$
$$0 \leq X(t) \leq (Y + 1 - 10\, Y D S_R) e^{-0.1t} + 10\, Y D S_R. \tag{45}$$

These bounds can be used to provide the guaranteed enclosure of the solution of the original systems state variables.

6 Simulation Results

In this section the Monte-Carlo method [1, 2] is compared with our methods using linear Lyapunov-like functions, Müller's Theorem and the midpoint method. Our approaches are providing guaranteed bounds for the given uncertain system. We achieve excellent results by intersecting the upper bounds from the methods using both Müller's Theorem and linear Lyapunov-like functions.

In Fig. 3 and Fig. 4 the upper bound provided by linear Lyapunov-like functions is much higher than any of the Monte-Carlo simulations. This comes from the constrained optimization, where we ask for the best upper bound at $t \to \infty$ and the simulation results are computed only up to 100 time units. Nevertheless the upper bound provided by Müller's Theorem is bounding both state variables as tight as possible.

The simulation results shown in Fig. 5 represent the substrate concentration. In this simulation the upper bound provided by linear Lyapunov-like functions and Müller's Theorem has the same value starting at approximately 200 time units. Comparing this to the Monte-Carlo simulation with the highest values it is obvious that there is no better upper bound than the given one. In Fig. 6 one can see the simulation of the bacteria concentration up to 400 time

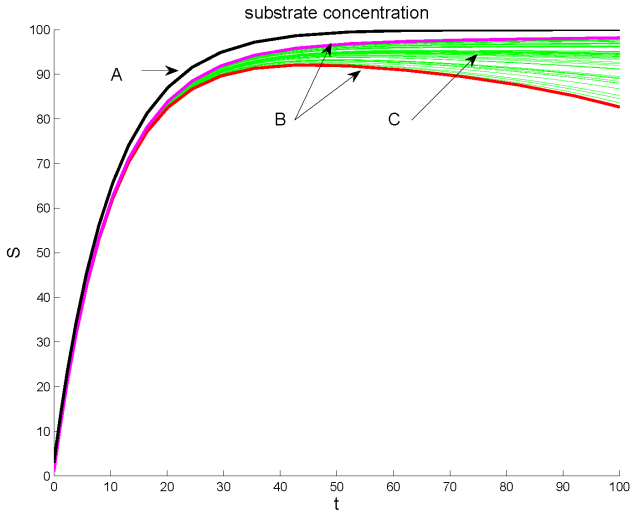


Figure 3. Simulations of the substrate concentration up to 100 time units
A: according to linear Lyapunov-like functions
B: corresponding to Müller's Theorem
C: provided by the Monte-Carlo method

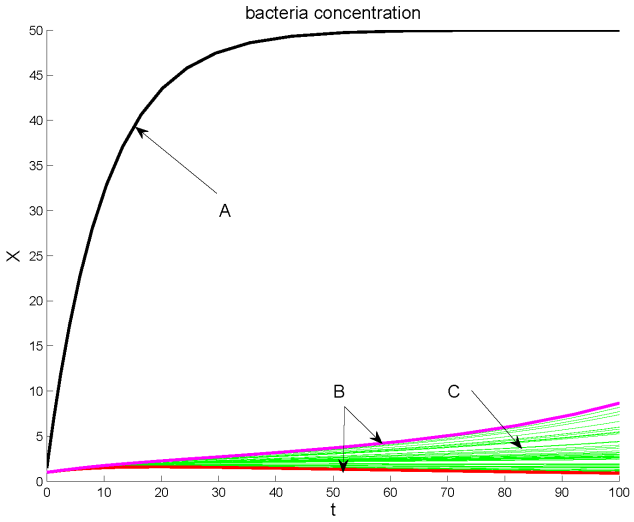


Figure 4. Simulations of the bacteria concentration up to 100 time units
A: according to linear Lyapunov-like functions
B: corresponding to Müller's Theorem
C: provided by the Monte-Carlo method

units. In this simulation the upper bound of Müller's Theorem is unstable and diverging. This is a common problem and a drawback of this method but by intersecting this result with the upper bound provided by linear Lyapunov-like functions, the resulting set includes all possible variations of the uncertain parameters and simultaneously represents guaranteed bounds.

7 Conclusions

In this paper we proposed an algorithm for a guaranteed simulation of nonlinear systems with uncertain parameters, which uses linear Lyapunov-like functions. To achieve optimal bounds for $t \to \infty$ we applied constrained optimization. A short overview on the algorithm and an application have been given. Moreover, we applied a second method for bounding uncertain nonlinear systems, which is using Müller's Theorem. A third approach was given with a midpoint method which exhibits overestimation. We got excellent bounds by intersecting the results of the first two mentioned methods. We have implemented the algorithms in Matlab [15] to perform the simulations. But these outputs does not provide validated simulation results, because the Matlab ODE solvers does not mention roundoff errors. There are some methods which provide validated results. One of these methods is presented by Lohner [5] and called the AWA-algorithm.

Using our methods guaranteed simulations of nonlinear systems with uncertain parameters are computed. This was demonstrated for a nonlinear system based on a bioreactor. The future work is to combine these approaches with other methods based on interval arithmetic [11, 12]. Especially the midpoint method needs to be improved to get tighter bounds.

References

[1] J.M. Hammersly, D.C. Handscomb, "Monte Carlo Methods", John Wiley & Sons, 1964.

[2] H. Kahn, "Applications of Monte Carlo", *Atomic Energy Commission Report-3259, April*, 1956.

[3] W. Walter, "Differential- und Integralungleichungen", Springer-Verlag, Berlin, 1964.

[4] M. Müller, "Über die Eindeutigkeit der Integrale eines Systems gewöhnlicher Differenzialgleichungen und die Konvergenz einer Gattung von Verfahren zur Approximation dieser Integrale", Sitzungsbericht Heidelberger Akad. d. Wiss., 1927.

[5] R. Lohner, "Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen", PhD-Thesis, Universität Karlsruhe, 1988.

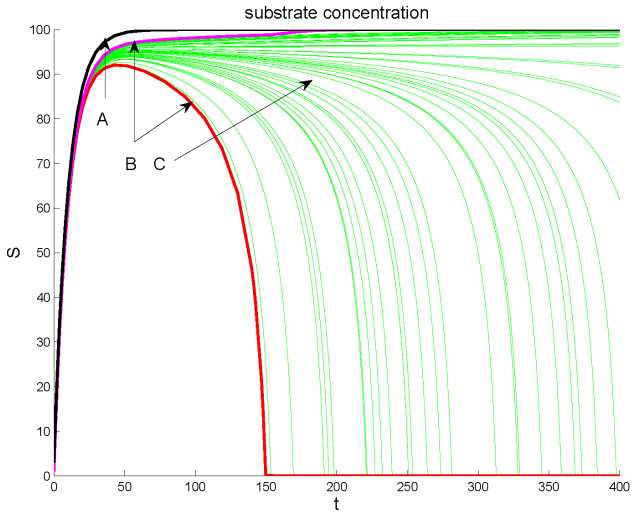


Figure 5. Simulations of the substrate concentration up to 400 time units
A: according to linear Lyapunov-like functions
B: corresponding to Müller's Theorem
C: provided by the Monte-Carlo method

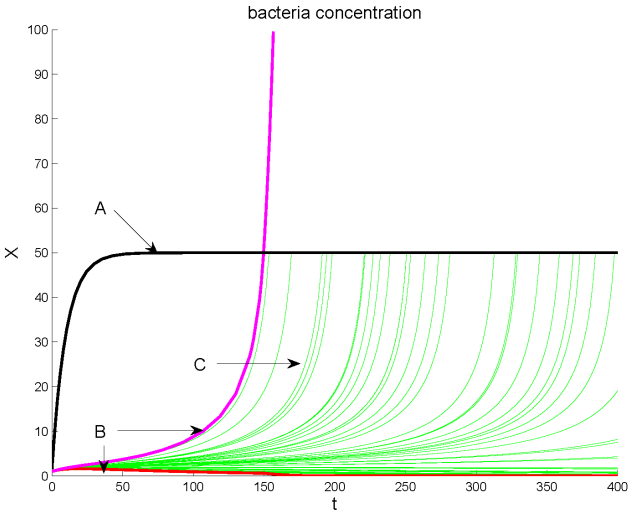


Figure 6. Simulations of the bacteria concentration up to 400 time units
A: according to linear Lyapunov-like functions
B: corresponding to Müller's Theorem
C: provided by the Monte-Carlo method

[6] T. H. Gronwall, "Note on the derivatives with respect to a parameter of the solutions of a system of differential equations", Ann. of Math. 20, pp. 292 - 296, 1918/19.

[7] G. Alefeld, J. Herzberger, "Introduction To Interval Computations", Academic Press, New York, 1983.

[8] E. Hansen, "Global Optimization Using Interval Analysis'", Marcel Dekker, New York, 1992.

[9] B. Tibken, M. Gennat, "Simulation of Nonlinear Systems with Uncertain Parameters Using Linear Lyapunov-like Functions", Proceedings of MCS 2004, pp. 83-88, Krakow, Poland.

[10] B. Tibken, M. Gennat, "A Novel Method of Nonlinear System-Simulation with Uncertain Parameters Providing Guaranteed Bounds", Proceedings of ACC 2005, pp. 709-713, Portland, OR, USA.

[11] B. Tibken, A. Bulach, J. Heeks, E.P. Hofer, "Neue Intervallarithmetische Methoden zur Garantierten Parameterschätzung", Statusseminar, Technische Anwendungen von Erkenntnissen der Nichtlinearen Dynamik, Frankfurt, 1999.

[12] H. Aschemann, A. Rauh, M. Kletting, E. P. Hofer, M. Gennat, B. Tibken, "Interval Analysis and Nonlinear Control of Wastewater Plants with Parameter Uncertainty", Proceedings of IFAC World Congress 2005, Prague, Czech Republic, 2005.

[13] O. Didrit, L. Jaulin, M. Kieffer, E. Walter,"Applied Interval Analysis", Springer-Verlag, London, 2001.

[14] D.W. Jordan, P. Smith, "Nonlinear Ordinary Differential Equations", Clarendon Press, Oxford, 1977.

[15] L. Shampine, M. Reichelt, "The Matlab ODE Suite", *SIAM J. Sci. Comput., 81*, **1**, pp. 1-22, 1997.

[16] U. Jeppson, "A General Description of the IAQW Activated Sludge Model 1", Lund Institute of Technology, Lund, Sweden.

[17] M. Köhne, "Analyse und Regelung biologischer Abwasserreinigungsprozesse in Kläranlagen", Automatisierungstechnik 46, at 5/98, R. Oldenbourg Verlag, 1998.

[18] B. Tibken, "Rechnergestützter Beobachterentwurf für Bilineare Systeme", PhD-Thesis, TU Hamburg-Harburg, 1991.

[19] C. Posten, personal communication.

On the Approximation of Linear AE-Solution Sets

Alexandre Goldsztejn*
University of California, Irvine
Irvine CA, USA
agoldy@ics.uci.edu

Gilles Chabert
INRIA
Sophia-Antipolis, France
Gilles.Chabert@sophia.inria.fr

Abstract

When considering systems of equations, it often happens that parameters are known with some uncertainties. This leads to continua of solutions that are usually approximated using the interval theory. A wider set of useful situations can be modeled if one allows furthermore different quantifications of the parameters in their domains. In particular, quantified solution sets where universal quantifiers are constrained to precede existential quantifiers are called AE-solution sets.

A state of the art on the approximation of linear AE-solution sets in the framework of generalized intervals (intervals whose bounds are not constrained to be ordered increasingly) is presented in a new and unifying way. Then two new generalized interval operators dedicated to the approximation of quantified linear interval systems are proposed and investigated.

1 Introduction

One of the most fundamental application of interval analysis is to allow one dealing with uncertain parameters (see [12, 15, 8] for some introductions to interval analysis). While a system of equations may have a discrete number of solutions given a value of its parameters, it is likely to have a manifold of solutions when parameters are constrained to belong to some intervals. When the problem to be solved consists of finding the solutions of systems of equations, considering interval domains for parameters gives rise to the definition of *united solution sets*:

$$\Sigma(f,\mathbf{a},\mathbf{b}) := \{x\in\mathbb{R}^n | \exists a\in\mathbf{a},\ \exists b\in\mathbf{b},\ f(a,x)=b\}, \quad (1)$$

where vectorial notations are used to obtain a compact expression (i.e. $f:\mathbb{R}^p\times\mathbb{R}^n\to\mathbb{R}^m$). In the special case where f is linear, the parameters $(a_i)_{i\in\{1,\ldots,p\}}$ are arranged into a matrix $A := (A_{ij})_{i,j\in\{1,\ldots,n\}}$ while an interval matrix $\mathbf{A} := (\mathbf{A}_{ij})_{i,j\in\{1,\ldots,n\}}$ is used to provide the interval domains, leading to the notation

$$\Sigma(\mathbf{A},\mathbf{b}) := \{x\in\mathbb{R}^n | \exists A\in\mathbf{A},\ \exists b\in\mathbf{b},\ Ax=b\}. \quad (2)$$

An important part of interval analysis is dedicated to the approximation of united solution sets, and in particular of linear united solution sets. Methods such as the interval Gauss-Seidel and Krawczyk operators, the interval Gauss elimination (cf. [15] and references therein), the Hansen-Bliek algorithm (cf. [16] and references therein), etc., are widely used to build some outer approximations of linear united solution sets.

A wider set of useful situations can be modeled by allowing different quantifiers for parameters. A general quantification of parameters is only little studied today in the framework of numerical analysis[1]. Constraining universal quantifiers to precede existential ones offers a compromise between the modeling power and the difficulty of the problems met in the study of the solution set. Such solution sets are called *AE-solution sets*[2] (cf. [28, 4, 3] for the approximation of linear and non-linear AE-solution sets and [9] for some applications). While interval analysis is well fitted to the approximation of united solution sets, it has been demonstrated that the formalism of generalized intervals (i.e. interval whose bounds are not constrained to be ordered) is the right framework for the approximation of AE-solution sets.

A state of the art of the methods dedicated to the approximation of linear AE-solution sets using generalized intervals is first proposed. This state of the art introduces a new convention for the representation of quantifiers using generalized intervals, allowing to homogenize the approximation of united solution sets in the framework of classical interval analysis and the study of AE-solution sets in the framework

*This work was mainly done during the author's thesis at the University of Sophia Antipolis and his post-doctoral study at the University of Central Arkansas.

[1] The quantifier elimination, with its well-known limitations, is a formal method which allows studying general quantifications (cf. [2]).

[2] This denomination coming from the succession of universal (All) quantifiers preceding existential (Exists) ones.

of generalized intervals. Then two new generalized interval operators dedicated to the outer approximation of linear AE-solution sets are presented.

2 Generalized intervals

Generalized intervals are intervals whose bounds are not constrained to be ordered, for example $[-1,1]$ and $[1,-1]$ are generalized intervals. They have been introduced in [17, 10] (see also [11, 28]) so as to improve the algebraic and the order structures of intervals. The set of generalized intervals is denoted by $\mathbb{KR}$ and is divided into three subsets:

- The set of *proper intervals* with bounds ordered increasingly. These proper intervals are identified with classical intervals. The set of proper intervals is denoted $\mathbb{IR} := \{[a,b] \mid a \leq b\}$. *Strictly* proper intervals satisfy $a < b$.
- The set of *improper intervals* with bounds ordered decreasingly. It is denoted by $\overline{\mathbb{IR}} := \{[a,b] \mid a \geq b\}$. *Strictly* improper intervals satisfy $a > b$.
- The set of *degenerated intervals* $\{[a,b] \mid a = b\} = \mathbb{IR} \cap \overline{\mathbb{IR}}$. Degenerated intervals are identified to reals.

Therefore, from a set of reals $\{x \in \mathbb{R} | a \leq x \leq b\}$ with $a \leq b$, one can build the two generalized intervals $[a,b]$ and $[b,a]$. It will be convenient to switch from one to the other keeping the underlying set of reals $\{x \in \mathbb{R} | a \leq x \leq b\}$ unchanged. To this purpose, the following three operations are introduced:

- the *dualization* is defined by $\mathrm{dual}\,[a,b] = [b,a]$;
- the *proper projection* is defined by $\mathrm{pro}\,[a,b] = [\min\{a,b\}, \max\{a,b\}]$;
- the *improper projection* is defined by $\mathrm{imp}\,[a,b] = [\max\{a,b\}, \min\{a,b\}]$.

The generalized intervals are partially ordered by an inclusion which extends the inclusion of classical intervals. Given two generalized intervals $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ and $\mathbf{y} = [\underline{\mathbf{y}}, \overline{\mathbf{y}}]$, the inclusion is defined by

$$\mathbf{x} \subseteq \mathbf{y} \iff \underline{\mathbf{y}} \leq \underline{\mathbf{x}} \wedge \overline{\mathbf{x}} \leq \overline{\mathbf{y}}. \tag{3}$$

For example, $[-1,1] \subseteq [-1.1,1.1]$ (this matches the set inclusion), $[1.1,-1.1] \subseteq [1,-1]$ (the inclusion between the underlying sets of reals is reversed for improper intervals) and $[2,0.9] \subseteq [-1,1]$. The latter case is known to provide the following interpretation: given $\mathbf{x}, \mathbf{y} \in \mathbb{IR}$, $\mathbf{x} \cap \mathbf{y} = \emptyset$ is equivalent to $\mathrm{dual}\,\mathbf{x} \subseteq \mathbf{y}$ (which is also equivalent to $\mathrm{dual}\,\mathbf{y} \subseteq \mathbf{x}$). As degenerated intervals are identified to reals, if $\mathbf{x}$ is proper then $x \in \mathbf{x} \iff x \subseteq \mathbf{x}$. On the other hand, if $\mathbf{x}$ is not proper then for all $x \in \mathbb{R}$ the inclusion $x \subseteq \mathbf{x}$ is false (hence a characterization of a set E of the form $x \in E \Longrightarrow x \subseteq \mathbf{x}$ means that $E \subseteq \mathbf{x}$ in the case where $\mathbf{x}$ is proper, while $E = \emptyset$ in the case where $\mathbf{x}$ is not proper). From this inclusion are defined generalized interval meet and join operation: $[a,b] \wedge [c,d] := [\max\{a,c\}, \min\{b,d\}]$ and $\vee := [\min\{a,c\}, \max\{b,d\}]$. Note that it is important to use different symbols to differentiate these latter operations with the union and intersection of classical intervals: e.g. $[-1,1] \cap [2,3] = \emptyset$ while $[-1,1] \wedge [2,3] = [2,1]$.

The generalized interval arithmetic (also called Kaucher arithmetic) extends the classical interval arithmetic. When only proper intervals are involved, this arithmetic coincides with the interval arithmetic: for $\mathbf{x}, \mathbf{y} \in \mathbb{IR}$ one has $\mathbf{x} \circ \mathbf{y} = \{x \circ y \in \mathbb{R} \mid x \in \mathbf{x}, y \in \mathbf{y}\}$. When proper and improper intervals are involved, some new operations are introduced. The expressions of the generalized interval addition and subtraction are the same as their classical counterparts:

$$[a,b] + [c,d] = [a+c, b+d], \tag{4}$$
$$[a,b] - [c,d] = [a-d, b-c]. \tag{5}$$

The expression of the generalized interval multiplication is given in Table 1 and Table 2, where

$$\begin{aligned} \mathcal{P} &= \{\mathbf{x} \in \mathbb{KR} | 0 \leq \underline{\mathbf{x}} \wedge 0 \leq \overline{\mathbf{x}}\} \\ -\mathcal{P} &= \{\mathbf{x} \in \mathbb{KR} | 0 \geq \underline{\mathbf{x}} \wedge 0 \geq \overline{\mathbf{x}}\} \\ \mathcal{Z} &= \{\mathbf{x} \in \mathbb{KR} | \underline{\mathbf{x}} \leq 0 \leq \overline{\mathbf{x}}\} \\ \mathrm{dual}\,\mathcal{Z} &= \{\mathbf{x} \in \mathbb{KR} | \underline{\mathbf{x}} \geq 0 \geq \overline{\mathbf{x}}\}. \end{aligned}$$

Table 1. Generalized interval multiplication

$\mathbf{x} \cdot \mathbf{y}$	$\mathbf{y} \in \mathcal{P}$	$\mathbf{y} \in \mathcal{Z}$
$\mathbf{x} \in \mathcal{P}$	$[\underline{\mathbf{x}}\,\underline{\mathbf{y}}, \overline{\mathbf{x}}\,\overline{\mathbf{y}}]$	$[\overline{\mathbf{x}}\,\underline{\mathbf{y}}, \overline{\mathbf{x}}\,\overline{\mathbf{y}}]$
$\mathbf{x} \in \mathcal{Z}$	$[\underline{\mathbf{x}}\,\overline{\mathbf{y}}, \overline{\mathbf{x}}\,\overline{\mathbf{y}}]$	$[\min\{\underline{\mathbf{x}}\,\overline{\mathbf{y}}, \overline{\mathbf{x}}\,\underline{\mathbf{y}}\}, \max\{\underline{\mathbf{x}}\,\underline{\mathbf{y}}, \overline{\mathbf{x}}\,\overline{\mathbf{y}}\}]$
$\mathbf{x} \in -\mathcal{P}$	$[\underline{\mathbf{x}}\,\overline{\mathbf{y}}, \overline{\mathbf{x}}\,\underline{\mathbf{y}}]$	$[\underline{\mathbf{x}}\,\overline{\mathbf{y}}, \underline{\mathbf{x}}\,\underline{\mathbf{y}}]$
$\mathbf{x} \in \mathrm{dual}\,\mathcal{Z}$	$[\underline{\mathbf{x}}\,\underline{\mathbf{y}}, \overline{\mathbf{x}}\,\underline{\mathbf{y}}]$	0

The generalized interval division $\mathbf{x}/\mathbf{y}$ is defined for intervals $\mathbf{x}$ and $\mathbf{y}$ which satisfy $0 \notin (\mathrm{pro}\,\mathbf{y})$. It can be characterized by $\mathbf{x}/\mathbf{y} = \mathbf{x} \times (1/\mathbf{y})$, where $1/\mathbf{y} = [1/\overline{\mathbf{y}}, 1/\underline{\mathbf{y}}]$.

The generalized interval arithmetic has better algebraic properties than the classical interval arithmetic: the addition in $\mathbb{KR}$ is a group. The opposite of an interval $\mathbf{x}$ is $-\mathrm{dual}\,\mathbf{x}$, i.e.,

$$\mathbf{x} + (-\mathrm{dual}\,\mathbf{x}) = \mathbf{x} - \mathrm{dual}\,\mathbf{x} = [0,0]. \tag{6}$$

Table 2. Generalized interval multiplication

$\mathbf{x}\cdot\mathbf{y}$	$\mathbf{y}\in -\mathcal{P}$	$\mathbf{y}\in \mathrm{dual}\,\mathcal{Z}$
$\mathbf{x}\in\mathcal{P}$	$[\overline{\mathbf{x}}\,\underline{\mathbf{y}}, \underline{\mathbf{x}}\,\overline{\mathbf{y}}]$	$[\underline{\mathbf{x}}\,\underline{\mathbf{y}}, \underline{\mathbf{x}}\,\overline{\mathbf{y}}]$
$\mathbf{x}\in\mathcal{Z}$	$[\overline{\mathbf{x}}\,\underline{\mathbf{y}}, \underline{\mathbf{x}}\,\underline{\mathbf{y}}]$	0
$\mathbf{x}\in -\mathcal{P}$	$[\overline{\mathbf{x}}\,\overline{\mathbf{y}}, \underline{\mathbf{x}}\,\underline{\mathbf{y}}]$	$[\overline{\mathbf{x}}\,\overline{\mathbf{y}}, \overline{\mathbf{x}}\,\underline{\mathbf{y}}]$
$\mathbf{x}\in \mathrm{dual}\,\mathcal{Z}$	$[\overline{\mathbf{x}}\,\overline{\mathbf{y}}, \underline{\mathbf{x}}\,\overline{\mathbf{y}}]$	$[\max\{\underline{\mathbf{x}}\,\underline{\mathbf{y}}, \overline{\mathbf{x}}\,\overline{\mathbf{y}}, \min\{\underline{\mathbf{x}}\,\overline{\mathbf{y}}, \overline{\mathbf{x}}\,\underline{\mathbf{y}}\}\}]$

The multiplication in $\mathbb{KR}$ restricted to generalized intervals whose proper projection does not contain 0 is also a group. The inverse of such a generalized interval $\mathbf{x}$ is $1/(\mathrm{dual}\,\mathbf{x})$, i.e.,

$$\mathbf{x}\cdot(1/\mathrm{dual}\,\mathbf{x}) = \mathbf{x}/(\mathrm{dual}\,\mathbf{x}) = [1,1]. \tag{7}$$

Although addition and multiplication in $\mathbb{KR}$ are associative, they are not distributive. The addition and multiplication in $\mathbb{KR}$ are linked by the following distributivity law, called the *conditional distributivity* (see [29, 18, 28]). Whatever are $\mathbf{x},\mathbf{y},\mathbf{z}\in\mathbb{KR}$,

$$\mathbf{x}\cdot\mathbf{y}+(\mathrm{imp}\,\mathbf{x})\cdot\mathbf{z} \;\subseteq\; \mathbf{x}\cdot(\mathbf{y}+\mathbf{z}) \;\subseteq\; \mathbf{x}\cdot\mathbf{y}+(\mathrm{pro}\,\mathbf{x})\cdot\mathbf{z}. \tag{8}$$

The three following particular cases will be of practical interest in this paper:

- *Subdistributivity*: if $\mathbf{x}\in\mathbb{IR}$ then $\mathbf{x}\cdot(\mathbf{y}+\mathbf{z})\subseteq\mathbf{x}\cdot\mathbf{y}+\mathbf{x}\cdot\mathbf{z}$.
- *Superdistributivity*: if $\mathbf{x}\in\overline{\mathbb{IR}}$ then $\mathbf{x}\cdot(\mathbf{y}+\mathbf{z})\supseteq\mathbf{x}\cdot\mathbf{y}+\mathbf{x}\cdot\mathbf{z}$.
- *Distributivity*: if $x\in\mathbb{R}$ then $x\cdot(\mathbf{y}+\mathbf{z})=x\cdot\mathbf{y}+x\cdot\mathbf{z}$.

Another useful property of the generalized interval arithmetic is its monotonicity with respect to inclusion: whatever are $\circ\in\{+,\cdot,-,\div\}$ and $\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}'\in\mathbb{KR}$,

$$\mathbf{x}\subseteq\mathbf{x}'\wedge\mathbf{y}\subseteq\mathbf{y}' \implies (\mathbf{x}\circ\mathbf{y})\subseteq(\mathbf{x}'\circ\mathbf{y}'). \tag{9}$$

More specifically the following equivalence holds:

$$\mathbf{x}\subseteq\mathbf{x}' \iff \mathbf{x}+\mathbf{y}\subseteq\mathbf{x}'+\mathbf{y}. \tag{10}$$

Finally, generalized interval vectors $\mathbf{x}\in\mathbb{KR}^n$ and generalized interval matrices $\mathbf{A}\in\mathbb{KR}^{n\times n}$ together with their additions and multiplications are defined similarly to their real and classical interval counterparts. The operations pro and dual are applied componentwise to these objects.

3 Linear AE-solution sets

AE-solution sets generalize united solution sets allowing different quantification of the parameters in their interval domains, with the constraint that universal quantifiers precede existential quantifiers. For example, given $\mathbf{A}\in\mathbb{IR}^{2\times2}$ and $\mathbf{b}\in\mathbb{IR}^2$, the following solution set is an AE-solution set:

$$\begin{aligned}\{x\in\mathbb{R}^2 | \forall A_{11}\in\mathbf{A}_{11}, \forall A_{21}\in\mathbf{A}_{21}, \forall A_{22}\in\mathbf{A}_{22},\\ \forall b_1\in\mathbf{b}_1, \exists A_{12}\in\mathbf{A}_{12}, \exists b_2\in\mathbf{b}_2, Ax=b\}.\end{aligned} \tag{11}$$

As quantifiers of the same kind commute, an AE-solution set is completely defined by one interval and one quantifier for each parameter A_{ij} and b_k. A useful representation of linear AE-solution sets is given in [28]: $\mathbf{A}^\alpha$ and $\mathbf{b}^\alpha$ are defined for $\alpha\in\{\forall,\exists\}$ as

$$(\mathbf{A}^\alpha)_{ij} = \begin{cases} \mathbf{A}_{ij} \text{ if } Q_{ij}=\alpha \\ 0 \text{ otherwise}\end{cases} \tag{12}$$

and

$$(\mathbf{b}^\alpha)_k = \begin{cases} \mathbf{b}_k \text{ if } Q_k=\alpha \\ 0 \text{ otherwise}\end{cases}, \tag{13}$$

where $Q_{ij},Q_k\in\{\exists,\forall\}$ are the quantifiers of the corresponding parameters. Thanks to these definitions, the general expression of a linear AE-solution can be given in the following way:

$$\begin{aligned}\{x\in\mathbb{R}^n | \forall A^\forall\in\mathbf{A}^\forall, \forall b^\forall\in\mathbf{b}^\forall, \exists A^\exists\in\mathbf{A}^\exists,\\ \exists b^\exists\in\mathbf{b}^\exists, (A^\forall+A^\exists)x=(b^\forall+b^\exists)\}.\end{aligned} \tag{14}$$

For example, the AE-solution set (11) can be defined by

$$\mathbf{A}^\exists=\begin{pmatrix}0 & \mathbf{A}_{12}\\ 0 & 0\end{pmatrix},\quad \mathbf{A}^\forall=\begin{pmatrix}\mathbf{A}_{11} & 0\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{pmatrix}, \tag{15}$$

$$\mathbf{b}^\exists=\begin{pmatrix}0 & \mathbf{b}_2\end{pmatrix}^{\mathrm{T}},\quad \mathbf{b}^\forall=\begin{pmatrix}\mathbf{b}_1 & 0\end{pmatrix}^{\mathrm{T}}. \tag{16}$$

Applied with (15) and (16), Definition (14) is obviously equivalent to (11). Finally, a second useful representation of AE-solution sets is introduced using generalized intervals. Each parameter has to be associated both an interval and a quantifier. Generalized intervals provide a very good representation of this couple of data. One associates a generalized interval to each parameter with the following interpretation:

- The proper projection of the generalized interval is the interval domain of this parameter.
- The quantifier associated to the parameter depends on the proper/improper quality of the generalized interval: $\mathbf{A}_{ij}\in\mathbb{IR} \iff Q_{ij}=\exists$ and hence $\inf\mathbf{A}_{ij} > \sup\mathbf{A}_{ij} \iff Q_{ij}=\forall$. Also, $\mathbf{b}_k\in\mathbb{IR} \iff Q_k=\exists$ and hence $\inf\mathbf{b}_k>\sup\mathbf{b}_k \iff Q_k=\forall$.

These two different definitions of linear AE-solution sets are obviously related by

$$\mathbf{A} = \mathbf{A}^{\exists} + \mathrm{dual}\,\mathbf{A}^{\forall} \quad (17)$$
$$\mathbf{b} = \mathbf{b}^{\exists} + \mathrm{dual}\,\mathbf{b}^{\forall}. \quad (18)$$

The convention chosen here to relate quantifiers and proper/improper qualities is different to the one used in [28]. The new convention has the great advantage of unifying the framework of AE-solution sets with the classical framework of united solution sets. The notation $\Sigma(\mathbf{A},\mathbf{b})$ can now be generalized to AE-solution sets: $\Sigma(\mathbf{A},\mathbf{b})$ is now defined for $\mathbf{A} \in \mathbb{KR}^{n\times n}$ and $\mathbf{b} \in \mathbb{KR}^n$ and corresponds to the AE-solution set obtained using the conventions stated above. In particular, if $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and $\mathbf{b} \in \mathbb{IR}^n$ then

$$\Sigma(\mathbf{A},\mathbf{b}) = \{x \in \mathbb{R}^n | \exists A \in \mathbf{A}, \exists b \in \mathbf{b}, Ax = b\}, \quad (19)$$

so the united solution set is defined as a particular AE-solution set in a homogeneous way. If $\mathbf{A} \in \overline{\mathbb{IR}}^{n\times n}$ and $\mathbf{b} \in \mathbb{IR}^n$ then

$$\Sigma(\mathbf{A},\mathbf{b}) = \{x \in \mathbb{R}^n | \forall A \in \mathrm{pro}\,\mathbf{A}, \exists b \in \mathbf{b}, Ax = b\} \quad (20)$$

and it is called a tolerable solution set. If $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and $\mathbf{b} \in \overline{\mathbb{IR}}^n$ then

$$\Sigma(\mathbf{A},\mathbf{b}) = \{x \in \mathbb{R}^n | \forall b \in \mathrm{pro}\,\mathbf{b}, \exists A \in \mathbf{A}, Ax = b\} \quad (21)$$

and it is called a controllable solution set. As another example, the AE-solution set (11) can be written $\Sigma(\mathbf{A}',\mathbf{b}')$ with

$$\mathbf{A}' = \begin{pmatrix} \mathrm{dual}\,\mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathrm{dual}\,\mathbf{A}_{21} & \mathrm{dual}\,\mathbf{A}_{22} \end{pmatrix} \quad (22)$$

and

$$\mathbf{b}' = \begin{pmatrix} \mathrm{dual}\,\mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}. \quad (23)$$

Remark. With these definitions, the new convention for the relationship between quantifiers and proper/improper qualities and the convention proposed in [28] are related by $\Sigma(\mathbf{A},\mathbf{b}) = \Xi(\mathrm{dual}\,\mathbf{A},\mathbf{b})$. This change of notation allows an homogenization between the classical interval analysis and the framework of generalized intervals dedicated to the approximation of AE-solution sets. As shown in [5], this homogenization is effective also in the context of non-linear united and AE-solution sets, where a generalized interval Newton operator dedicated to the approximation of non-linear AE-solution sets has been proposed.

Not only do generalized intervals provide us with a very convenient modeling language for AE-solution sets, but they are also a powerful analytical framework for their study. In particular, the following characterization theorem discovered and proved by Shary is of central importance. Its statement differs from the one proposed in [28] only by the new convention chosen for the relation between the quantifiers and the proper/improper qualities of generalized intervals. Also, the proof proposed here is reproduced from the one given in [28] using the new convention. This proof provides a interesting insight of the relationship between the group structure of the generalized interval addition and its interpretation in the context of AE-solution sets.

Theorem 1. *Let* $\mathbf{A} \in \mathbb{KR}^{n\times n}$ *and* $\mathbf{b} \in \mathbb{KR}^n$. *Then*

$$x \in \Sigma(\mathbf{A},\mathbf{b}) \iff (\mathrm{dual}\,\mathbf{A})x \subseteq \mathbf{b}. \quad (24)$$

Proof. By definition of an AE-solution set, $x \in \Sigma(\mathbf{A},\mathbf{b})$ is equivalent to

$$\forall A^{\forall} \in \mathbf{A}^{\forall}, \forall b^{\forall} \in \mathbf{b}^{\forall}, \exists A^{\exists} \in \mathbf{A}^{\exists}, \exists b^{\exists} \in \mathbf{b}^{\exists}, \quad (A^{\forall} + A^{\exists})x = b^{\forall} + b^{\exists}. \quad (25)$$

where $\mathbf{A}^{\forall}$, $\mathbf{A}^{\forall}$, $\mathbf{b}^{\forall}$ and $\mathbf{b}^{\forall}$ are defined as in (12) and (13), so $\mathbf{A} = \mathbf{A}^{\exists} + \mathrm{dual}\,\mathbf{A}^{\forall}$ and $\mathbf{b} = \mathbf{b}^{\exists} + \mathrm{dual}\,\mathbf{b}^{\forall}$. Now, $(A^{\forall} + A^{\exists})x = (b^{\forall} + b^{\exists})$ is equivalent to $A^{\forall}x - b^{\forall} = -A^{\exists}x + b^{\exists}$. Furthermore, obviously both $\mathbf{A}^{\forall}x - \mathbf{b}^{\forall} = \{A^{\forall}x - b^{\forall} | A^{\forall} \in \mathbf{A}^{\forall}, b^{\forall} \in \mathbf{b}^{\forall}\}$ and $-\mathbf{A}^{\exists}x + \mathbf{b}^{\exists} = \{-A^{\exists}x + b^{\exists} | A^{\exists} \in \mathbf{A}^{\exists}, b^{\exists} \in \mathbf{b}^{\exists}\}$ hold because each parameter has only one occurrence in the whole system. As a consequence, (25) is equivalent to

$$\mathbf{A}^{\forall}x - \mathbf{b}^{\forall} \subseteq -\mathbf{A}^{\exists}x + \mathbf{b}^{\exists}. \quad (26)$$

Now, adding $\mathrm{dual}\,(\mathbf{A}^{\exists}x) + \mathrm{dual}\,\mathbf{b}^{\forall}$ and using the group property of the generalized interval addition, one proves that (26) is equivalent to

$$\mathbf{A}^{\forall}x + \mathrm{dual}\,(\mathbf{A}^{\exists}x) \subseteq \mathbf{b}^{\exists} + \mathrm{dual}\,\mathbf{b}^{\forall}. \quad (27)$$

Finally, noticing that $\mathbf{A}^{\forall}x + \mathrm{dual}\,(\mathbf{A}^{\exists}x) = \mathbf{A}^{\forall}x + (\mathrm{dual}\,\mathbf{A}^{\exists})x = (\mathbf{A}^{\forall} + \mathrm{dual}\,\mathbf{A}^{\exists})x$, the last equality being a consequence of the distributivity of the addition and multiplication in $\mathbb{KR}$, one obtain the equivalent inclusion $(\mathrm{dual}\,\mathbf{A})x \subseteq \mathbf{b}$. □

The characterization (24) is an elegant generalization of the previously known characterizations in the special cases of united, tolerable and controllable solution sets:

- If $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and $\mathbf{b} \in \mathbb{IR}^n$ then $(\mathrm{dual}\,\mathbf{A})x \subseteq \mathbf{b}$ is equivalent to $\mathbf{A}x \cap \mathbf{b} \neq \emptyset$ which is the well-known characterization of united solution sets.
- If $\mathbf{A} \in \overline{\mathbb{IR}}^{n\times n}$ and $\mathbf{b} \in \mathbb{IR}^n$ then $(\mathrm{dual}\,\mathbf{A})x \subseteq \mathbf{b}$ is equivalent to $(\mathrm{pro}\,\mathbf{A})x \subseteq \mathbf{b}$ which is the well-known characterization of tolerable solution sets.
- If $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and $\mathbf{b} \in \overline{\mathbb{IR}}^n$ then $(\mathrm{dual}\,\mathbf{A})x \subseteq \mathbf{b}$ is equivalent to $(\mathrm{pro}\,\mathbf{b}) \subseteq \mathbf{A}x$ which is the well-known characterization of controllable solution sets.

As a direct consequence of Theorem 1, we have $\mathbf{A} \subseteq \mathbf{A}'$ and $\mathbf{b} \subseteq \mathbf{b}'$ imply $\Sigma(\mathbf{A},\mathbf{b}) \subseteq \Sigma(\mathbf{A}',\mathbf{b}')$. In particular $\Sigma(\mathbf{A},\mathbf{b}) \subseteq \Sigma(\mathrm{pro}\,\mathbf{A},\mathrm{pro}\,\mathbf{b})$ which states that universal quantifiers are more restrictive than existential ones.

4 Approximation of linear AE-solution sets

As in the classical interval theory for the approximation of linear united solution sets, the approximation of linear AE-solution sets can be done both through the characterization of the solutions of some auxiliary interval equation or using contracting interval operators. The following subsections give a survey of these methods. The use of the new convention relating the quantifiers and the proper/improper quality of generalized intervals allows us to give an homogenized presentation with the classical interval framework.

Throughout this section, $\mathbf{A} \in \mathbb{KR}^{n\times n}$ and $\mathbf{b} \in \mathbb{KR}^n$ are considered. They correspond to a linear AE-solution set $\Sigma(\mathbf{A}, \mathbf{b})$.

4.1 Auxiliary interval equations

This section presents three generalized interval equations whose solutions can be interpreted as some approximation of linear AE-solution sets. The technique which consists in solving some auxiliary interval equation to build approximation of AE-solution set is called the *formal algebraic approach* to AE-solution set approximation (see [24, 13, 23, 14, 26, 21, 22, 4]).

Remark. It must be noted that, in spite of the usefulness of the formal algebraic approach to AE-solution set approximation, no rigorous rounding process has yet been proposed to allow correctly rounded approximations.

4.1.1 Auxiliary interval equation dedicated to inner approximation

While inner approximation is not studied in the classical interval analysis of linear united solution sets, it arises naturally in the generalized interval analysis of linear AE-solution sets. The following theorem provides a way to build an inner approximation of some linear AE-solution set. The proof is reproduced from the one given by Shary in [28] but using the new convention relating the quantifiers and the proper/improper quality of generalized intervals.

Theorem 2. *Every proper solution $\mathbf{x} \in \mathbb{IR}^n$ of the interval equation*

$$(\operatorname{dual} \mathbf{A})\mathbf{x} = \mathbf{b} \tag{28}$$

is an inner approximation of $\Sigma(\mathbf{A}, \mathbf{b})$, i.e. $\mathbf{x} \subseteq \Sigma(\mathbf{A}, \mathbf{b})$.

Proof. Let a proper interval vector $\mathbf{x}$ be a formal solution of the equation (28) and $\tilde{x} \in \mathbf{x}$. Then, in view of the monotonicity of the generalized interval arithmetic, we have $(\operatorname{dual} \mathbf{A})\tilde{x} \subseteq (\operatorname{dual} \mathbf{A})\mathbf{x} = \mathbf{b}$, that is $x \in \Sigma(\mathbf{A}, \mathbf{b})$ by Theorem 1. □

Remark. Using the convention relating the quantifiers and the proper/improper quality of generalized intervals previously used in [28, 21, 4], the equation (28) is written $\mathbf{A}\mathbf{x} = \mathbf{b}$ in the latter papers. However, Kupriyanova gave in [13] the same expression and the same interpretation for the equation (28) but in the restricted case of united solution sets.

4.1.2 Auxiliary interval equations dedicated to outer approximation

Two generalized interval equations have been proposed for the outer approximation of linear AE-solution sets. The following theorem is proposed by Shary in [26] (see also [28]).

Theorem 3. *Suppose that $\rho(|I - \operatorname{pro} \mathbf{A}|) < 1$. Then, the following interval equation has an unique solution $\mathbf{x} \in \mathbb{KR}^n$:*

$$\mathbf{x} = (I - \mathbf{A})\mathbf{x} + \mathbf{b}. \tag{29}$$

Furthermore, if the solution $\mathbf{x}$ is proper then it contains $\Sigma(\mathbf{A}, \mathbf{b})$; if the solution $\mathbf{x}$ is not proper then $\Sigma(\mathbf{A}, \mathbf{b})$ is empty.

Equation (29) is written $\mathbf{x} = (I - \operatorname{dual} \mathbf{A})\mathbf{x} + \mathbf{b}$ in [28, 21, 4]. Thanks to the newly introduced convention relating the quantifiers and the proper/improper quality of generalized intervals, one can now see the similitude of this equation with the Krawczyk operator dedicated to linear united solution sets: in the case where $\mathbf{A}$ and $\mathbf{b}$ are proper, iterating the fixed point form of Equation (29) corresponds to the Krawczyk operator where the intersection with the last approximation would not be performed.

4.1.3 Solving generalized interval equations

Several ways can be used to solve the auxiliary interval equations presented in the previous section. The most simple, though very efficient, is to write them in a fixed point form and to iterate this fixed point form. If this iteration converges then its limit is a solution of the fixed point form, and hence of the original equation. A widely used fixed point form of (28) is

$$\bigwedge_{i\in\{1,\ldots,n\}} \mathbf{x}_i = \frac{1}{\operatorname{dual} \mathbf{A}_{ii}}\Big(\mathbf{b}_i - \sum_{\substack{j\in[1..n]\\ j\neq i}} \operatorname{dual}\,(\mathbf{A}_{ij}\mathbf{x}_j)\Big). \tag{30}$$

and is defined provided that $0 \notin \operatorname{pro} \mathbf{A}_{ii}$ for all $i \in \{1, \ldots, n\}$. It is proved in [4] that this fixed point iteration converges to the unique solution of (28) if $\operatorname{pro} \mathbf{A}$ is an H-matrix (see also [14]). Equation (29) is already in fixed point form. The fixed point iteration is proved to converge to the unique solution of (29) provided that $\rho(|I - \operatorname{pro} \mathbf{A}|) < 1$ (see [26, 28]).

Other resolution processes have been proposed: Shary has proposed in [23, 25] to immerse the interval equation to be solved in $\mathbb{R}^{2n}$ where a system of $2n$ real equations is obtained making explicit the expression of the generalized interval arithmetic. The resulting real system is not differentiable and a dedicated solving process have been proposed by Shary. Also, Sainz et al. have proposed in [21] to change the auxiliary interval equation to an optimization problem which can be handled using existing optimization softwares. Finally, the group structures of the generalized interval arithmetic allow in some situation to solve equations and systems of equations. This has been studied e.g. in [18].

4.2 Generalized interval operators

Only one generalized interval operator is available for the outer approximation of linear AE-solution sets: Shary has proposed a generalized interval Gauss-Seidel operator in [27] (see also [28]). It has the same interpretation as its classical interval counterpart, but extended to linear AE-solution sets. Using the newly introduced convention relating the quantifiers and the proper/improper quality of generalized intervals, Shary's generalized interval Gauss-Seidel operator now has the same expression as its classical counterpart.

Theorem 4. *Provided that* $0 \notin \operatorname{pro} \mathbf{A}_{ii}$ *for all* $i \in \{1,\dots,n\}$*, the generalized interval Gauss-Seidel operator* $\Gamma(\mathbf{A},\mathbf{b},\mathbf{x})$ *is defined by* $\Gamma(\mathbf{A},\mathbf{b},\mathbf{x}) := \mathbf{y}$ *with*

$$\mathbf{y}_i := \frac{1}{\mathbf{A}_{ii}}\Big(\mathbf{b}_i - \sum_{j<i} \mathbf{A}_{ij}\mathbf{y}_j - \sum_{j>i} \mathbf{A}_{ij}\mathbf{x}_j\Big), \qquad (31)$$

If $\Gamma(\mathbf{A},\mathbf{b},\mathbf{x})$ *is proper then* $\Sigma(\mathbf{A},\mathbf{b}) \cap \mathbf{x} \subseteq \Gamma(\mathbf{A},\mathbf{b},\mathbf{x})$. *Otherwise* $\Sigma(\mathbf{A},\mathbf{b}) \cap \mathbf{x} = \emptyset$.

Remark. Using the other convention relating the quantifiers and the proper/improper quality of generalized intervals, the generalized interval Gauss-Seidel operator is written $(1/\operatorname{dual} \mathbf{A}_{ii})\big(\mathbf{b}_i - \sum_{j<i}(\operatorname{dual} \mathbf{A}_{ij})\mathbf{y}_j - \sum_{j>i}(\operatorname{dual} \mathbf{A}_{ij})\mathbf{x}_j\big)$ in [28].

Theorem 4 is the typical situation where the newly introduced convention shows its usefulness: the classical and generalized interval Gauss-Seidel operators now share both their expression and their interpretation. Note however that the classical interval Gauss-Seidel operator can be applied in situations where $0 \in \operatorname{pro} \mathbf{A}_{ii}$ thanks to the use of an extended division while the generalized interval Gauss-Seidel operator proposed by Shary needs $0 \notin \operatorname{pro} \mathbf{A}_{ii}$ for all $i \in \{1,\dots,n\}$. This gap will be filled in Section 6.

Remark. The generalized interval Gauss-Seidel operator can be used with outer rounding, hence rigorously preserving the inclusion of the AE-solution set.

4.3 Preconditioning

Preconditioning consists in studying an auxiliary solution set where the approximation methods both keep their interpretation and have a better behavior. Two kinds of preconditioning have been proposed for linear AE-solution sets. First, Shary proved that

$$\Sigma(\mathbf{A},\mathbf{b}) \subseteq \Sigma(C\mathbf{A}, C\mathbf{b}), \qquad (32)$$

where $C \in \mathbb{R}^{n\times n}$ is regular, hence generalizing the preconditioning process used in the context of united solution sets. Indeed by (32), an outer approximation of $\Sigma(C\mathbf{A}, C\mathbf{b})$ is also an outer approximation of $\Sigma(\mathbf{A},\mathbf{b})$. While such a preconditioning process improves the stability of the Gauss-Seidel operator, it is even more important when using Theorem 3 where the condition $\rho(|I - \operatorname{pro} \mathbf{A}|) < 1$ is actually very restrictive. Using the midpoint inverse preconditioning, i.e. $C = (\operatorname{mid} \mathbf{A})^{-1}$, Theorem 3 can be applied with any interval matrix that satisfies $\operatorname{pro} \mathbf{A}$ is strongly regular (see [28] for more details).

Another preconditioning process dedicated to inner approximation has been proposed by the first author in [4]. There, the following inclusion is proved[3]:

$$\{\tilde{x} + Cu | u \in \Sigma(\operatorname{imp} \mathbf{A}C, b - \operatorname{dual} \mathbf{A}\tilde{x})\} \subseteq \Sigma(\mathbf{A},\mathbf{b}), \qquad (33)$$

where $C \in \mathbb{R}^{n\times n}$ is regular. As a consequence, if $\mathbf{u}$ is an inner approximation of the auxiliary AE-solution set $\Sigma(\operatorname{imp} \mathbf{A}C, b - \operatorname{dual} \mathbf{A}\tilde{x})$ then the following inclusion holds:

$$\{\tilde{x} + Cu | u \in \mathbf{u}\} \subseteq \Sigma(\mathbf{A},\mathbf{b}). \qquad (34)$$

Therefore, computing an inner approximation $\mathbf{u}$ of the auxiliary AE-solution set $\Sigma(\operatorname{imp} \mathbf{A}C, b - \operatorname{dual} \mathbf{A}\tilde{x})$ gives rise to an inner approximation of the original AE-solution set $\Sigma(\mathbf{A},\mathbf{b})$ under the form of a parallelepiped $\{\tilde{x} + Cu | u \in \mathbf{u}\}$. Inner approximation methods (in particular the fixed point iteration) often have a better behavior when applied to $\Sigma(\operatorname{imp} \mathbf{A}C, b - \operatorname{dual} \mathbf{A}\tilde{x})$ using e.g. $C = (\operatorname{mid} \mathbf{A})^{-1}$.

5 Generalized interval Krawczyk operator

Thanks to the new conventions used in the present paper, it is now clear that the auxiliary interval equation (29) is similar to the Krawczyk operator for linear united solution sets. It is therefore natural to introduce a generalized interval Krawczyk operator.

Theorem 5. *The generalized interval Krawczyk operator is defined by*

$$\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x}) := \Big((I - \mathbf{A})\mathbf{x} + \mathbf{b}\Big) \wedge \mathbf{x}, \qquad (35)$$

[3] The inclusion (33) is actually an obvious consequence of Proposition 4.1 and Proposition 4.2 of [4]

where an non-proper interval vector is interpreted as the emptyset. If $\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$ *is proper then* $\Sigma(\mathbf{A},\mathbf{b})\cap\mathbf{x}\subseteq\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$. *Otherwise* $\Sigma(\mathbf{A},\mathbf{b})\cap\mathbf{x}=\emptyset$.

Proof. Consider any $x\in\mathbf{x}\cap\Sigma(\mathbf{A},\mathbf{b})$. Then by Theorem 1 the inclusion $(\operatorname{dual}\mathbf{A})x\subseteq\mathbf{b}$ holds. The generalized interval distributivity gives rise to $(\operatorname{dual}\mathbf{A})x = x+(\operatorname{dual}\mathbf{A}-I)x$. Therefore $x+(\operatorname{dual}\mathbf{A}-I)x\subseteq\mathbf{b}$ holds. Now, adding $-\operatorname{dual}\big((\operatorname{dual}\mathbf{A}-I)x\big)=(I-\mathbf{A})x$ to each side of the inclusion gives rise to $x\subseteq(I-\mathbf{A})x+\mathbf{b}$. Finally the interval inclusion isotonicity proves $x\subseteq(I-\mathbf{A})\mathbf{x}+\mathbf{b}$ and hence $x\subseteq\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$ because $x\subseteq\mathbf{x}$. As a consequence, if $\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$ is proper then $x\in\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$. On the other hand if $\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$ is not proper then no x can satisfy $x\subseteq\mathbf{K}(\mathbf{A},\mathbf{b},\mathbf{x})$ and therefore $\Sigma(\mathbf{A},\mathbf{b})\cap\mathbf{x}$ is empty. □

The generalized interval Krawczyk operator presents several advantages on its generalized interval equation counterpart (29): first it can be applied with no restriction on $\mathbf{A}$. Second, an initial enclosure can be provided in order to enclose $\Sigma(\mathbf{A},\mathbf{b})\cap\mathbf{x}$. If no initial enclosure is available and if $\mathbf{A}$ is strongly regular then one can use $[-1,1]\,\langle\operatorname{pro}\mathbf{A}\rangle^{-1}\,|\operatorname{pro}\mathbf{b}|$ which is a enclosure of $\Sigma(\operatorname{pro}\mathbf{A},\operatorname{pro}\mathbf{b})$ (cf. [15]).

6 Improved generalized interval Gauss-Seidel operator

The generalized interval Gauss-Seidel (IGS in this section) operator presented in Section 4.2 cannot be applied if $0\in\operatorname{pro}\mathbf{A}_{ii}$ for some i. The situation was similar with the first version of classical IGS operator proposed in [20]. Latter, an improved classical IGS have been proposed in [7] which allows dealing with zero-containing intervals. This section presents an improved version of the generalized interval Gauss-Seidel operator that can deal with any generalized interval matrix, hence extending the ideas of [7] to linear AE-solution sets.

6.1 One dimensional case

Following the presentation given in [15] in the context of united solution sets, the improved generalized IGS operator is first defined in the case $n=1$. In this situation, the AE-solution set $\Sigma(\mathbf{a},\mathbf{b})$ is defined by two generalized intervals $\mathbf{a},\mathbf{b}\in\mathbb{KR}$. Given furthermore a proper interval $\mathbf{x}\in\mathbb{IR}$, the improved generalized IGS operator is defined as

$$\Gamma(\mathbf{a},\mathbf{b},\mathbf{x}) := \square\big(\Sigma(\mathbf{a},\mathbf{b})\bigcap\mathbf{x}\big). \tag{36}$$

The next proposition provides a computable expression of $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})$.

Proposition 1. *In the following expressions, an improper interval is interpreted as an emptyset.*

1. *If* $0\notin\operatorname{pro}\mathbf{a}$ *then* $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})=(\mathbf{b}/\mathbf{a})\bigwedge\mathbf{x}$.
2. *If* $0\subseteq\mathbf{a}$ *then define* E *as the open interval* $\big]\min\{0,\underline{\mathbf{b}}/\underline{\mathbf{a}},\overline{\mathbf{b}}/\overline{\mathbf{a}}\}\ ,\ \max\{0,\underline{\mathbf{b}}/\overline{\mathbf{a}},\overline{\mathbf{b}}/\underline{\mathbf{a}}\}\big[$[4]. *Then,* $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})=\square(\mathbf{x}\backslash E)$.
3. *If* $\mathbf{a}\subseteq 0$ *then* $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})=\big[\max\{\underline{\mathbf{b}}/\underline{\mathbf{a}},\overline{\mathbf{b}}/\overline{\mathbf{a}}\}\ ,\ \min\{\underline{\mathbf{b}}/\overline{\mathbf{a}},\overline{\mathbf{b}}/\underline{\mathbf{a}}\}\big]\bigwedge\mathbf{x}$.

Proof. Cf. [5] pages 105-107. □

Remark. As in [15], the expression of $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})$ is not detailed in the cases where $\underline{\mathbf{a}}=0$ or $\overline{\mathbf{a}}=0$. Also, one can check that when $0\in\mathbf{a}$ and $\mathbf{b}$ is proper, the expression proposed here coincides with the one proposed in [15] in the context of one dimensional linear united solution sets.

The next example illustrates Proposition 1.

Example 1. Consider $\mathbf{a}=[1,-1]$ and $\mathbf{b}=[-1,3]$ and the one dimensional AE-solution set $\Sigma(\mathbf{a},\mathbf{b})=$

$$\big\{x\in\mathbb{R}\mid\big(\forall a\in[-1,1]\big)\big(\exists b\in[-1,3]\big)\big(ax=b\big)\big\}\,. \tag{37}$$

Determining this simple AE-solution set can be done graphically. Fix a value of a in $[-1,1]$. Then the set $\big\{x\in\mathbb{R}\mid\big(\exists b\in[-1,3]\big)\big(ax=b\big)\big\}$ can be easily determined: it is the set of solutions of the equation $ax=b$ for some $b\in[1,3]$ (it is plotted on the left hand side graphic of Figure 1 for $a=1$). Therefore, in order to obtain the set of $x\in\mathbb{R}$ that satisfies $\big(\exists b\in[-1,3]\big)\big(ax=b\big)$ for any $a\in[-1,1]$, it remains to intersect all the strips obtained for $a\in[-1,1]$. Three of these strips and the resulting intersection are plotted on the right hand side graph where we can see that $\Sigma(\mathbf{a},\mathbf{b})=[-1,1]$. It can be noted that the AE-solution set is bounded while $0\in\operatorname{pro}\mathbf{a}$. Now, given an initial $\mathbf{x}=[-\infty,\infty]$, the generalized IGS $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})$ leads to $\big[\max\{\underline{\mathbf{b}}/\underline{\mathbf{a}},\overline{\mathbf{b}}/\overline{\mathbf{a}}\}\ ,\ \min\{\underline{\mathbf{b}}/\overline{\mathbf{a}},\overline{\mathbf{b}}/\underline{\mathbf{a}}\}\big]$, which is equal to $[-1,1]$. This is indeed equal to $\Sigma(\mathbf{a},\mathbf{b})$.

Link with a generalized interval division

Proposition 1 can be used to extend the generalized interval division to cases where the numerator's proper projection contains zero: in these cases $\mathbf{b}/\mathbf{a}$ is defined by

$$\begin{array}{l}\big]-\infty,\min\{0,\underline{\mathbf{b}}/\underline{\mathbf{a}},\overline{\mathbf{b}}/\overline{\mathbf{a}}\}\big]\\ \qquad\bigcup\ \big[\max\{0,\underline{\mathbf{b}}/\overline{\mathbf{a}},\overline{\mathbf{b}}/\underline{\mathbf{a}}\},+\infty\big[\end{array}\ \text{if } 0\subseteq\mathbf{a} \tag{38}$$

$$\big[\max\{\underline{\mathbf{b}}/\underline{\mathbf{a}},\overline{\mathbf{b}}/\overline{\mathbf{a}}\}\ ,\ \min\{\underline{\mathbf{b}}/\overline{\mathbf{a}},\overline{\mathbf{b}}/\underline{\mathbf{a}}\}\big]\ \text{ if } \mathbf{a}\subseteq 0. \tag{39}$$

[4] This open interval is well defined because the first bound is always lower or equal than the second. Note furthermore that $]0,0[=\emptyset$ while $\square\emptyset=\emptyset$.

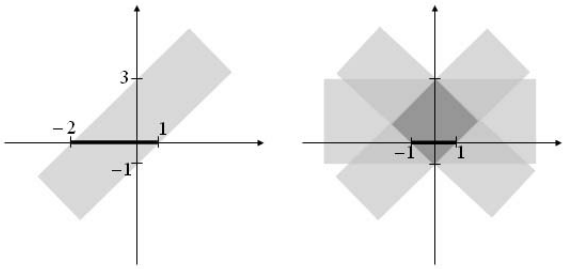


Figure 1.

Then, the generalized IGS expression can be written $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x}) = \square(1/\mathbf{a} \cap \mathbf{x})$, where an improper $1/\mathbf{a}$ is interpreted as an empty set for the intersection with $\mathbf{x}$. An other extension of the generalized interval division has been proposed in [10] and further studied in [19]. The study of the relationship between these two extensions remains to be conducted.

6.2 The general case

A n dimensional generalized IGS can now be constructed: consider $\mathbf{A} \in \mathbb{KR}^{n\times n}$, $\mathbf{b} \in \mathbb{KR}^n$ and $\mathbf{x} \in \mathbb{IR}^n$. Then the generalized IGS $\Gamma(\mathbf{A},\mathbf{b},\mathbf{x})$ is defined in the following way: for $i \in \{1,\ldots,n\}$

$$\Gamma(\mathbf{A},\mathbf{b},\mathbf{x})_i := \Gamma\Big(\mathbf{A}_{ii}\,,\, \mathbf{b}_i - \sum_{j<i} \mathbf{A}_{ij}\mathbf{y}_j - \sum_{j>i} \mathbf{A}_{ij}\mathbf{x}_j\,,\, \mathbf{x}_i\Big), \tag{40}$$

where $\Gamma(\mathbf{a},\mathbf{b},\mathbf{x})$ is defined in Subsection 6.1. The next theorem provides the interpretation of this improved generalized IGS.

Theorem 6. *Let* $\mathbf{A} \in \mathbb{KR}^{n\times n}$, $\mathbf{b} \in \mathbb{KR}^{n\times n}$ *and* $\mathbf{x} \in \mathbb{IR}^{n\times n}$. *Then*

$$\Sigma(\mathbf{A},\mathbf{b}) \bigcap \mathbf{x} \subseteq \Gamma(\mathbf{A},\mathbf{b},\mathbf{x}) \subseteq \mathbf{x} \tag{41}$$

Proof. Cf. [5] pages 107-108. □

The usefulness of the improved generalized IGS operator is illustrated by the following example. As in [3], both interval Gauss-Seidel operators are applied not only on the diagonal entries of the matrix but on all entries, leading to $n \times n$ applications of the one dimensional operator instead of only n applications.

Example 2. Consider

$$\mathbf{A} = \begin{pmatrix} 1 & [-1,1] \\ 2 & [1,-1] \end{pmatrix} \quad , \quad \mathbf{b} = \begin{pmatrix} [1,-1] \\ [-1,1] \end{pmatrix}. \tag{42}$$

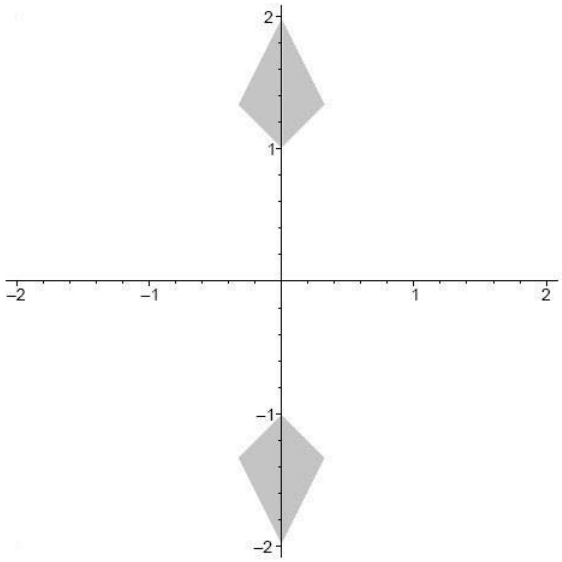


Figure 2.

The AE-solution set $\Sigma(\mathbf{A},\mathbf{b})$ is displayed on Figure 2. Note that this AE-solution set is bounded while $\mathrm{pro}\,\mathbf{A}$ is not regular which is not possible in the context of united solution sets. The initial box is set to $([-10,10],[0,20])$. The interval Gauss-Seidel without the improvement proposed in this paper leads to the contracted box $([-1,1],[0,20])$. The improved Gauss-Seidel leads to $([-1,1],[0,4])$ and hence provides an additional contraction.

7 Conclusion

AE-solution sets generalize the united-solution sets allowing different quantifications of parameters, the universal quantifiers preceding the existential ones. They allow to model useful situations. It has become clear that generalized intervals are the right framework for the study of AE-solution sets. Thanks to the proposition of a new convention for the association of a quantifier to a generalized interval, the studies of united solution sets and AE-solution sets have been homogenized. This has allowed to propose a generalized interval Krawczyk operator and an extension of Shary's generalized interval Gauss-Seidel to interval matrices with zero-containing intervals on the diagonal.

The interval Gauss elimination and Hansen-Bliek algorithm have not been considered in this paper. They have been investigated by the authors in [6] and [1] respectively. However, their generalization to AE-solution sets is not as direct as the techniques presented in the present paper, and still some work remains to be conducted in this direction.

References

[1] G. Chabert and A. Goldsztejn. Extension of the Hansen-Bliek Method to Right-Quantified Linear Systems. *Reliab. Comp.*, 13(4):325–349, 2007.

[2] G. Collins. Quantifier elimination by cylindrical algebraic decomposition–twenty years of progress. *In Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23, 1998.

[3] A. Goldsztejn. A Branch and Prune Algorithm for the Approximation of Non-Linear AE-Solution Sets. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1650–1654.

[4] A. Goldsztejn. A Right-Preconditioning Process for the Formal-Algebraic Approach to Inner and Outer Estimation of AE-solution Sets. *Reliab. Comp.*, 11(6):443–478, 2005.

[5] A. Goldsztejn. *Définition et Applications des Extensions des Fonctions Réelles aux Intervalles Généralisés*. PhD thesis, Université de Nice-Sophia Antipolis, 2005.

[6] A. Goldsztejn and G. Chabert. A Generalized Interval LU Decomposition for the Solution of Interval Linear Systems. In *Proceedings NM&A 2006*, Borovets, Bulgaria, August 2006.

[7] E. Hansen and S. Sengupta. Bounding solutions of systems of equations using interval analysis. *BIT*, 21:203–211, 1981.

[8] B. Hayes. A Lucid Interval. *American Scientist*, 91(6):484–488, 2003.

[9] M. Jirstand. Nonlinear control system design by quantifier elimination. *Journal of Symbolic Computation*, 24(2):137–152, 1997.

[10] E. Kaucher. *Uber metrische und algebraische Eigenschaften einiger beim numerischen Rechnen auftretender Raume*. PhD thesis, Karlsruhe, 1973.

[11] E. Kaucher. Interval Analysis in the Extended Interval Space $\mathbb{IR}$. *Computing*, Suppl. 2:33–49, 1980.

[12] R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath*, Bulletin 2(1):95–112, 1996.

[13] L. Kupriyanova. Inner estimation of the united solution set of interval linear algebraic system. *Reliab. Comp.*, 1(1):15–31, 1995.

[14] S. Markov, E. Popova, and C. Ullrich. On the solution of linear algebraic equations involving interval coefficients. *Iterative Methods in Linear Algebra, IMACS Series on Computational and Applied Mathematics*, 3:216–225, 1996.

[15] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, Cambridge, 1990.

[16] A. Neumaier. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. *Reliab. Comp.*, 5:131–136, 1999.

[17] H.-J. Ortolf. Eine Verallgemeinerung der Intervallarithmetik. *Geselschaft fuer Mathematik und Datenverarbeitung, Bonn*, 11:1–71, 1969.

[18] E. Popova. Multiplication distributivity of proper and improper intervals. *Reliab. Comp.*, 7(2):129–140, 2001.

[19] E. D. Popova. Extended Interval Arithmetic in IEEE Floating-Point Environment. *Reliab. Comp.*, 4:100–129, 1994.

[20] F. Ris. *Interval analysis and applications to linear algebra*. PhD thesis, Oxford, 1972.

[21] M. Sainz, E. Gardenyes, and L. Jorba. Formal Solution to Systems of Interval Linear or Non-Linear Equations. *Reliab. Comp.*, 8(3):189–211, 2002.

[22] M. Sainz, E. Gardenyes, and L. Jorba. Interval Estimations of Solutions Sets to Real-Valued Systems of Linear or Non-Linear Equations. *Reliab. Comp.*, 8(4):283–305, 2002.

[23] S. Shary. Algebraic Approach to the Interval Linear Static Identification, Tolerance and Control Problems, or One More Application of Kaucher Arithmetic. *Reliab. Comp.*, 2:3–33, 1996.

[24] S. Shary. Algebraic Solutions to Interval Linear Equations and their Application. In *Proceedings of the IMACS - GAMM International Symposium on Numerical Methods and Error Bounds, Oldenburg, 1995*. Editors: G. Alefeld and J. Herzberger, 1996.

[25] S. Shary. Algebraic approach in the "outer problem" for interval linear equations. *Reliab. Comp.*, 3:103–135, 1997.

[26] S. Shary. Outer estimation of generalized solution sets to interval linear systems. *Reliab. Comp.*, 5:323–335, 1999.

[27] S. Shary. Interval Gauss-Seidel Method for Generalized Solution Sets to Interval Linear Systems. *Reliab. Comp.*, 7:141–155, 2001.

[28] S. Shary. A new technique in systems analysis under interval uncertainty and ambiguity. *Reliab. Comp.*, 8:321–418, 2002.

[29] SIGLA/X group. Modal intervals. *Reliab. Comp.*, 7:77–111, 2001.

Rigorous Inner Approximation of the Range of Functions

Alexandre Goldsztejn*
University of California, Irvine
Irvine CA, USA
agoldy@ics.uci.edu

Wayne Hayes
University of California, Irvine
Irvine CA, USA
wayne@ics.uci.edu

Abstract

A basic problem of interval analysis is the computation of a superset of the image of an interval by a function, called an outer enclosure. Here we consider the computation of an inner enclosure, which is a subset *of the image. Inner approximations are harder than the outer ones in general: proving that a box is inside the image is equivalent to proving existence of solutions for a collection of systems of equations. Based on this remark, a new construction of the inner approximation is proposed that is particularly efficient for small domains. Then, it is shown than one can apply these ideas in the context of ordinary differential equations, hence providing some tools of potential interest for the theory of shadowing in dynamical systems.*

1 Introduction

Interval analysis is naturally used to construct subsets (also called *outer approximations*, *enclosures*) of the range of real functions, i.e., intervals that rigorously contain the image of some domain by the function (see [18, 11] for an introduction to interval analysis). Constructing inner approximations of the range of a real function—i.e., intervals that are rigorously contained inside the image of some domain by the function—needs additional developments. Although many authors have investigated the construction of inner approximations in the case of functions $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ (see [27, 15] and references therein), a lot of work remains in the case of vector-valued functions $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$.

There is a direct relationship between inner approximation and existence theorems for systems of equations. Indeed, if one is able to build an inner approximation $\mathcal{I} \subseteq \mathrm{range}(f, \mathbf{x})$ then $0 \in \mathcal{I}$ is obviously a sufficient condition for $(\exists x \in \mathbf{x})(f(x) = 0)$. On the other hand, it seems that many existence theorems can naturally be used to build some inner approximation: defining $g(x) = f(x) - z$, $\mathcal{I} \subseteq \mathrm{range}(f, \mathbf{x})$ holds if $g(x)$ is proved to have a zero in $\mathbf{x}$ for all $z \in \mathcal{I}$ (this technique seems to be first proposed in [5] and further used in [9]). Hence, a parameterized version of an existence theorem would be useful.

This observation leads to a branch and prune algorithm [9], hence allowing inner approximation of the range of vector valued functions $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. Such bisection algorithms are well-suited when the dimension of the image is small, or when the evaluation of the function and its derivatives are inexpensive. Otherwise, it may be useful to build an inner approximation of a small interval domain in one computation, in a similar way as the mean-value extension to intervals provides an outer approximation in one computation. The present paper provides such a construction. That is, when the interval domain is small enough (roughly speaking when the mean-value extension computes an accurate outer approximation) one will be able to build both inner and outer approximations of the image of the interval domain by a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. To this end, we need only an interval enclosure of the image of the center of the domain, and an interval Lipschitz matrix for the whole domain. Hence, the inner approximation comes for almost no additional computational cost beyond that of the outer approximation.

Some potential applications of such inner approximations are foreseen in the context of shadowing dynamical systems [13]. Therefore, some techniques dedicated to the computation of some Lipschitz interval matrix of the solution operator to the IVP with respect to the initial condition are finally presented. In this context, the computation of a Lipschitz interval matrix involves the evaluation of the Taylor expansion (of dimension $n^2 + n$), and is hence very expensive. Computing an inner approximation for the same cost as the outer approximation is therefore valuable in this context.

2 Interval Analysis

We assume familiarity with basic interval analysis [23, 1, 28, 17, 16, 11]. Following [19], intervals, interval vectors

*This work was done mainly while the first author was a post-doctoral fellow at the University of Central Arkansas.

and interval matrices are denoted by boldface letters.

Definition 1 (Lipschitz interval matrix (LIM)). Let $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ be a continuous function and $\mathbb{Y} \subseteq \mathbb{R}^n$. The interval matrix $\mathbf{J} \in \mathbb{IR}^{n\times m}$ is a LIM for ϕ and $\mathbb{Y}$ if and only if

$$\big(\forall x, y \in \mathbb{Y}\big)\big(\,\phi(x) - \phi(y) \in \mathbf{J}\cdot(x-y)\,\big), \qquad (1)$$

or equivalently

$$\big(\forall x, y \in \mathbb{Y}\big)\big(\exists J \in \mathbf{J}\big)\big(\,\phi(x) - \phi(y) = J\cdot(x-y)\,\big). \quad (2)$$

The existence of a LIM $\mathbf{J}$ for ϕ and $\mathbb{Y}$ implies that the function is Lipschitz inside $\mathbb{Y}$ with constant $||\mathbf{J}||$. If ϕ is continuously differentiable then

$$\frac{\partial\phi(x)}{\partial x}\Big|_y, \qquad (3)$$

i.e., the partial derivative of $\phi(x)$ with respect to x evaluated at y, is denoted by $\phi'(y)$ when no confusion is possible. In this case, it is well known that any interval matrix that contains $\{\phi'(x) \mid x \in \mathbb{Y}\}$ is a LIM for ϕ and $\mathbb{Y}$ [28]. Let us also recall that a LIM gives rise to the following interval enclosure (usually called the *mean-value extention*):

$$\mathrm{range}\big(f, \mathbf{x}\big) \subseteq f(\tilde{x}) + \mathbf{J}\cdot(\mathbf{x} - \tilde{x}), \qquad (4)$$

where $\mathbf{J}$ is a LIM for f and $\mathbf{x} \in \mathbb{IR}^n$, and $\tilde{x} \in \mathbf{x}$.

3 Inner and Outer Approximations of the Range of Real Functions

This section presents an inner approximation process for functions $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. We first construct box approximations (i.e., axis-aligned interval vectors). Then, more general parallelepipeds are used to allow inner approximation in a wider set of situations.

3.1 Box approximations

The well-known Poincaré-Miranda theorem (cf. [30][1]) is a generalization of the Intermediate Value Theorem to continuous functions $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. Interval analysis is well suited for a rigorous application of the Poincaré-Miranda theorem. Theorem 1 below, proposed by Frommer et al. in [4] following the idea proposed by Moore and Kioustelidis in [24] **CITATION?**, is a computationally efficient corollary of the Poincaré-Miranda theorem.

Definition 2. Define $\mathbf{x}^{(i-)} \in \mathbb{IR}^n$ and $\mathbf{x}^{(i+)} \in \mathbb{IR}^n$ as follows: $\mathbf{x}_k^{(i\pm)} = \mathbf{x}_k$ for $k \neq i$ and either $\mathbf{x}_i^{(i-)} = \inf \mathbf{x}_i$ or $\mathbf{x}_i^{(i+)} = \sup \mathbf{x}_i$ for $k = i$.

[1] A scanned version of [30] is available at http://www.goldsztejn.com/downloads.htm

Hence, $\mathbf{x}^{(i\pm)}$ is a pair of opposite faces of the interval box $\mathbf{x}$.

Theorem 1 (Frommer et. al.[4]). *Let $\mathbf{x} \in \mathbb{IR}^n$ be an interval vector and $f : \mathbf{x} \longrightarrow \mathbb{R}^n$ be a continuous function. Consider a real vector $\tilde{x} \in \mathbf{x}$. Let $\mathbf{J} \in \mathbb{IR}^{n\times n}$ be a LIM for f and $\mathbf{x}$. Suppose that, for all $i \in \{1,\ldots,n\}$, both*

$$\sup\big(f_i(\tilde{x}) + \mathbf{J}_{i:}\cdot(\mathbf{x}^{(i-)} - \tilde{x})\big) \leq 0 \qquad (5)$$

and

$$0 \leq \inf\big(f_i(\tilde{x}) + \mathbf{J}_{i:}\cdot(\mathbf{x}^{(i+)} - \tilde{x})\big), \qquad (6)$$

where $\mathbf{J}_{i:} \in \mathbb{IR}^{1\times n}$ is the i^{th} row of $\mathbf{J}$. Then there exists $x \in \mathbf{x}$ such that $f(x) = 0$.

The following definition allows Theorem 1 to be conveniently reformulated to aid the forthcoming Corollary 1.

Definition 3. Let $\mathbf{x}, \mathbf{z} \in \mathbb{IR}^n$, $\tilde{x} \in \mathbf{x}$ and $\mathbf{J} \in \mathbb{IR}^{n\times n}$. Define for $i \in \{1,\ldots,n\}$ define the reals

$$\underline{\mathbf{b}}_i := \sup\big(\mathbf{z}_i + \mathbf{J}_{i:}\cdot(\mathbf{x}^{(i-)} - \tilde{x})\big) \qquad (7)$$

$$\overline{\mathbf{b}}_i := \inf\big(\mathbf{z}_i + \mathbf{J}_{i:}\cdot(\mathbf{x}^{(i+)} - \tilde{x})\big). \qquad (8)$$

Define, $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ and $\mathcal{O}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ in the following way:

$$\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) := \begin{cases} \mathbf{b} & \text{if } \forall i \in \{1,\ldots,n\}, \underline{\mathbf{b}}_i \leq \overline{\mathbf{b}}_i \\ \emptyset & \text{otherwise} \end{cases} \qquad (9)$$

$$\mathcal{O}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) := \mathbf{z} + \mathbf{J}(\mathbf{x} - \tilde{x}). \qquad (10)$$

if $\underline{\mathbf{b}}_i \leq \overline{\mathbf{b}}_i$ holds for all $i \in \{1,\ldots,n\}$ then $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) := \mathbf{b}$ (where $\mathbf{b}_i = [\underline{\mathbf{b}}_i, \overline{\mathbf{b}}_i]$). Otherwise $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) := \emptyset$, while $\mathcal{O}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) := \mathbf{z} + \mathbf{J}(\mathbf{x} - \tilde{x})$.

Then Theorem 1 can be stated in the following way: if $0 \in \mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ then there exists $x \in \mathbf{x}$ such that $f(x) = 0$. The following corollary of Theorem 1 allows one to build inner approximations of the image of $\mathbf{x}$ by f. Both inner and outer approximations are considered in Corollary 1 in a homogenized way.

Corollary 1. *Let $\mathbf{x} \in \mathbb{IR}^n$ be an interval vector and $f : \mathbf{x} \longrightarrow \mathbb{R}^n$ be a continuous function. Consider a real vector $\tilde{x} \in \mathbf{x}$ and an interval vector $\mathbf{z} \in \mathbb{IR}^n$ that contains $f(\tilde{x})$. Let $\mathbf{J} \in \mathbb{IR}^{n\times n}$ be a LIM for f and $\mathbf{x}$. Then*

$$\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big) \subseteq \mathrm{range}\big(f, \mathbf{x}\big) \subseteq \mathcal{O}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big). \qquad (11)$$

Proof. The second inclusion is obvious from (10) and the definition of $\mathbf{J}$. Now consider the first inclusion. If $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ is empty then the first inclusion trivially holds. Now suppose that $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ is not empty and consider any fixed $z \in \mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ and define the auxiliary function $g(x) := f(x) - z$. We just have to prove that g has

a zero in $\mathbf{x}$. The interval matrix $\mathbf{J}$ is obviously also a LIM for g and $\mathbf{x}$. By the definition of $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ the following two inequalities hold $\forall i \in \{1, \ldots, n\}$:

$$\begin{array}{c} \sup\big(\mathbf{z}_i + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i-)} - \tilde{x})\big) \\ \leq z_i \leq \\ \inf\big(\mathbf{z}_i + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i+)} - \tilde{x})\big). \end{array} \tag{12}$$

Because $f_i(\tilde{x}) \in \mathbf{z}_i$, we have $\inf \mathbf{z}_i \leq f_i(\tilde{x}) \leq \sup \mathbf{z}_i$ and therefore,

$$\begin{array}{c} \sup\big(f_i(\tilde{x}) + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i-)} - \tilde{x})\big) \\ \leq z_i \leq \\ \inf\big(f_i(\tilde{x}) + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i+)} - \tilde{x})\big). \end{array} \tag{13}$$

Subtracting the fixed number z_i and noticing that $f_i(x) - z_i = g_i(x)$, we obtain

$$\begin{array}{c} \sup\big(g_i(\tilde{x}) + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i-)} - \tilde{x})\big) \\ \leq 0 \leq \\ \inf\big(g_i(\tilde{x}) + \mathbf{J}_{i:} \cdot (\mathbf{x}^{(i+)} - \tilde{x})\big). \end{array} \tag{14}$$

As this holds for all $i \in \{1, \ldots, n\}$, we can apply Theorem 1 so that g has a zero in $\mathbf{x}$, which concludes the proof. □

Remark 1. Corollary 1 allows building a box inner approximation. Such inner approximations were first computed using a mean-value extension to generalized intervals (i.e. interval whose bounds are not constrained to be increasingly ordered) in [6, 8]. Surprisingly, the inner approximations built in [6, 8] are exactly identical to the one built using Corollary 1. This coincidence has been the basis for the comparison between existence theorems proposed in [7]. Corollary 1 does not need the introduction of generalized intervals, which is an important advantage w.r.t. the techniques proposed in [6, 8].

It is worth stressing that the rigorous outer rounding for interval arithmetic leads to rigorous inner approximation thanks to Theorem 1. $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ can be nonempty only if the interval matrix $\mathbf{J}$ is close enough to the identity matrix. Formally, $\mathbf{J}$ has to be an interval H-matrix[2] (cf. [7]). This is very restrictive and preconditioning is the usual way to weaken this restrictive necessary condition. The next section shows how the preconditioning usually associated to Theorem 1 leads to parallelepiped approximations when Corollary 1 is used.

3.2 Parallelepiped Approximations

Throughout this section, we consider a function $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and a set $\mathbb{Y}_0$ wrapped between two parallelepipeds

$$\mathbb{A}_0 := \{M_0 u \mid u \in \mathbf{a}_0\}, \tag{15}$$

$$\mathbb{B}_0 := \{M_0 u \mid u \in \mathbf{b}_0\}, \tag{16}$$

[2] A H-matrix is a diagonally dominant matrix which has been scaled.

where $\mathbf{a}_0, \mathbf{b}_0 \in \mathbb{IR}^n$ and $M \in \mathbb{R}^{n \times n}$ (i.e., $\mathbb{A}_0 \subseteq \mathbb{Y}_0 \subseteq \mathbb{B}_0$). We aim to construct two parallelepipeds

$$\mathbb{A}_1 := \{M_1 u \mid u \in \mathbf{a}_1\}, \tag{17}$$

$$\mathbb{B}_1 := \{M_1 u \mid u \in \mathbf{b}_1\}, \tag{18}$$

such that

$$\mathbb{A}_1 \subseteq \mathbb{Y}_1 \subseteq \mathbb{B}_1 \text{ where } \mathbb{Y}_1 := \mathrm{range}\big(\phi, \mathbb{Y}_0\big). \tag{19}$$

We require the matrices M_0 and M_1 to be nonsingular. The only description of $\mathbb{Y}_0$ we have are the wrapping parallelepipeds $\mathbb{A}_0$ and $\mathbb{B}_0$. Therefore, we are going to compute $\mathbb{A}_1$ and $\mathbb{B}_1$ such that

$$\mathbb{A}_1 \subseteq \mathrm{range}\big(\phi, \mathbb{A}_0\big) \text{ and } \mathrm{range}\big(\phi, \mathbb{B}_0\big) \subseteq \mathbb{B}_1, \tag{20}$$

which obviously implies (19). To apply Corollary 1, we need to work in some auxiliary bases, where parallelepipeds will be represented by their characteristic boxes. This is represented in the following diagram, where the left hand side represents the actual action of ϕ, and the right hand side represents the action of ϕ in the auxiliary bases, leading to the function $\psi := M_1^{-1} \circ \phi \circ M_0$ (matrices are identified with the linear mappings they represent).

$$\begin{array}{ccc} \mathbb{A}_0 \subseteq \mathbb{Y}_0 \subseteq \mathbb{B}_0 & \xleftarrow{M_0} & \mathbf{a}_0 \subseteq \widetilde{\mathbb{Y}}_0 \subseteq \mathbf{b}_0 \\ \Big\downarrow \phi & & \Big\downarrow \psi \\ \mathbb{A}_1 \subseteq \mathbb{Y}_1 \subseteq \mathbb{B}_1 & \xleftarrow{M_1} & \mathbf{a}_1 \subseteq \widetilde{\mathbb{Y}}_1 \subseteq \mathbf{b}_1 \end{array}$$

In the auxiliary bases, we have $\mathbf{a}_0 \subseteq \widetilde{\mathbb{Y}}_0 \subseteq \mathbf{b}_0$, where $\widetilde{\mathbb{Y}}_0 := \big\{M_0^{-1} y \mid y \in \mathbb{Y}_0\big\}$, and we need to construct $\mathbf{a}_1$ and $\mathbf{b}_1$ such that $\mathbf{a}_1 \subseteq \widetilde{\mathbb{Y}}_1 \subseteq \mathbf{b}_1$, where $\widetilde{\mathbb{Y}}_1 := \mathrm{range}\big(\psi, \widetilde{\mathbb{Y}}_0\big)$. So we can use Corollary 1 and obtain $\mathbf{a}_1$ and $\mathbf{b}_1$ such that

$$\mathbf{a}_1 \subseteq \mathrm{range}\big(\psi, \mathbf{a}_0\big) \subseteq \widetilde{\mathbb{Y}}_1 \subseteq \mathrm{range}\big(\psi, \mathbf{b}_0\big) \subseteq \mathbf{b}_1. \tag{21}$$

Therefore, coming back in the original basis (through M_1) the interval vectors $\mathbf{a}_1$ and $\mathbf{b}_1$ give rise to the inner and outer approximations $\mathbb{A}_1$ and $\mathbb{B}_1$ that satisfy (20) and eventually (19). The matrix M_1, which gives the parallelepipeds $\mathbb{A}_1$ and $\mathbb{B}_1$ their shape, can be chosen arbitrarily. Some possible choices are proposed in Subsection 3.3.

To apply Corollary 1 with the function ψ, we need a LIM for this function and $\mathbf{b}_0$. The next proposition aids the computation of such a LIM.

Proposition 1. *Let $\mathbf{J}$ be a LIM for $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and $\mathbb{B} := \{Mu \mid u \in \mathbf{b}\}$, where $M \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{IR}^n$. Consider $M' \in \mathbb{R}^{n \times n}$. Then, $\mathbf{J}' := M' \cdot \mathbf{J} \cdot M$ is a LIM for $\psi := M' \circ \phi \circ M$ and $\mathbf{b}$.*

Proof. For any $u, v \in \mathbf{b}$, we have

$$\begin{aligned} \psi(u) - \psi(v) &= M'\phi(Mu) - M'\phi(Mv) \quad (22) \\ &= M'\big(\phi(Mu) - \phi(Mv)\big). \quad (23) \end{aligned}$$

Now, both $Mu, Mv \in \mathbb{B}$, and $\mathbf{J}$ is a LIM for ϕ and $\mathbb{B}$, so there exists $J \in \mathbf{J}$ such that $\phi(Mu) - \phi(Mv) = J(Mu - Mv)$. Therefore

$$\begin{aligned} \psi(u) - \psi(v) &= M'J(Mu - Mv) \quad (24) \\ &= M'JM(u - v). \quad (25) \end{aligned}$$

Finally, as $\mathbf{J}' \supseteq \{M'JM \mid J \in \mathbf{J}\}$, we can find $\tilde{J} \in \mathbf{J}'$ such that $\psi(u) - \psi(v) = \tilde{J}(u - v)$, so $\mathbf{J}'$ is a LIM for ψ and $\mathbf{b}$. □

The construction of the parallelepiped inner and outer approximations is formalized by the following theorem.

Theorem 2. *With the notations defined at the beginning of the section, consider $a \in \mathbf{a}_0$ and $b \in \mathbf{b}_0$ and define*

$$\begin{aligned} \mathbf{a}_1 &:= \mathcal{I}\big(\mathbf{a}_0, a, \Psi_a, \mathbf{J}'\big) \quad (26) \\ \mathbf{b}_1 &:= \mathcal{O}\big(\mathbf{b}_0, b, \Psi_b, \mathbf{J}'\big), \quad (27) \end{aligned}$$

with $\mathbf{J}' := M_1^{-1}\mathbf{J}M_0$, where $\mathbf{J}$ is a LIM for ϕ and $\mathbb{B}_0$, and Ψ_a and Ψ_b are interval vectors that contain respectively $\psi(a)$ and $\psi(b)$ with $\psi := M_1^{-1} \circ \phi \circ M_0$. Then

$$\mathbb{A}_1 \subseteq \mathrm{range}\big(\phi, \mathbb{Y}_0\big) \subseteq \mathbb{B}_1. \quad (28)$$

Proof. Proposition 1 proves that $\mathbf{J}'$ is a LIM for ψ and $\mathbf{b}_0$ (and therefore also for ψ and $\mathbf{a}_0$ because $\mathbf{a}_0 \subseteq \mathbf{b}_0$). From the definitions of $\mathbf{a}_1$ and $\mathbf{b}_1$, we can apply Corollary 1 which proves

$$\mathbf{a}_1 \subseteq \mathrm{range}\big(\psi, \mathbf{a}_0\big) \text{ and } \mathrm{range}\big(\psi, \mathbf{b}_0\big) \subseteq \mathbf{b}_1. \quad (29)$$

By the definition of ψ, we have both

$$\begin{aligned} \mathrm{range}\big(\psi, \mathbf{a}_0\big) &= \mathrm{range}\big(M_1^{-1}\phi, \mathbb{A}_0\big) \quad (30) \\ \mathrm{range}\big(\psi, \mathbf{b}_0\big) &= \mathrm{range}\big(M_1^{-1}\phi, \mathbb{B}_0\big). \quad (31) \end{aligned}$$

Therefore, applying the linear mapping M_1 to each inclusion (29), we obtain $\mathbb{A}_1 \subseteq \mathrm{range}\big(\phi, \mathbb{A}_0\big)$ and $\mathrm{range}\big(\phi, \mathbb{B}_0\big) \subseteq \mathbb{B}_1$, which concludes the proof by (20). □

All one needs in order to apply Theorem 2 is:

(i) Interval enclosures Ψ_a and Ψ_b of $\psi(a)$ and $\psi(b)$. In general, it will be easy to compute some interval enclosures Φ_a and Φ_b of $\phi(M_0 a)$ and $\phi(M_0 b)$. Then one can use $\Psi_a := M_1^{-1}\Phi_a$ and $\Psi_b := M_1^{-1}\Phi_b$.

(ii) A LIM $\mathbf{J}$ for ϕ and $\mathbb{B}_0$. It will be easier to compute a LIM $\mathbf{J}$ for ϕ and $M_0\mathbf{b}_0$ (the interval vector $M_0\mathbf{b}_0$ being the smallest interval vector that contains $\mathbb{B}_0$ so $\mathbb{B}_0 \subseteq M_0\mathbf{b}_0$).

In the case where $\phi = \phi_K \circ \cdots \circ \phi_2 \circ \phi_1$, Theorem 2 can be applied inductively in the following way. Let $\mathbb{A}_k$ and $\mathbb{B}_k$ be respectively inner and outer approximations of the image of $\mathbb{Y}_0$ by the function $\phi_k \circ \cdots \circ \phi_2 \circ \phi_1$, with $k < K$. Then, knowing $\mathbb{A}_k$ and $\mathbb{B}_k$, we can compute $\mathbb{A}_{k+1}$ and $\mathbb{B}_{k+1}$ using Theorem 2. This process obviously preserves the interpretations of the approximations, hence leading to $\mathbb{A}_K \subseteq \mathrm{range}\big(\phi_K \circ \cdots \circ \phi_2 \circ \phi_1, \mathbb{Y}_0\big) \subseteq \mathbb{B}_K$.

Remark 2. In the case where $\mathbb{A}_k = \emptyset$ for some $k \in \{1, \ldots, K\}$ then $\mathbb{A}_k = \emptyset$ for $k \in \{k, \ldots, K\}$. So, only the outer approximation component of Theorem 2 is used.

3.3 On the Choice of the Parallelepiped Characteristic Matrix

The characteristic matrix M_1 decides the shape of the parallelepiped approximations $\mathbb{A}_1$ and $\mathbb{B}_1$. Any M_1 will satisfy the theorems, but some choices give better approximations than others.

As mentioned at the end of Subsection 3.1, $\mathcal{I}\big(\mathbf{x}, \tilde{x}, \mathbf{z}, \mathbf{J}\big)$ can be nonempty only if $\mathbf{J}$ is an interval H-matrix. The LIM of ψ that will be used is $M_1^{-1}(\mathbf{J}M_0)$. As a consequence of the theory of strongly regular interval matrices [28], $\mathbf{J}M_0$ has to be strongly regular and one should choose $M_1 = \mathrm{mid}\,(\mathbf{J}M_0)$. This relates in an interesting way on one hand the midpoint inverse preconditioning usually used with Theorem 1, and on the other hand the shape of the parallelepiped approximations that are built using Corollary 1.

Finally, if $\mathbf{J}M_0$ is not strongly regular, then no inner approximation is possible. If an outer approximation has to be computed, then $M_1 = \mathrm{mid}\,(\mathbf{J}M_0)$ may not be an appropriate choice, and outer approximation certainly needs a better choice for the auxiliary basis M_1. In this case, the QR-factorization method, widely used in the context of ordinary differential equation solving, can be used for a better choice of M_1 [31, 26].

3.4 A Linear Mapping Example

Consider the rotation $f(x) = R_\theta x$ where R_θ is the angle θ rotation matrix. We choose $\theta = \pi/10$. We use two Lipschitz interval matrices. The first is obtained by evaluating the derivative of f with interval arithmetic:

$$\mathbf{J} := \begin{pmatrix} 0.95105651629515_3^4 & -0.309016994374947_8^7 \\ 0.309016994374947_7^8 & 0.95105651629515_3^4 \end{pmatrix}, \quad (32)$$

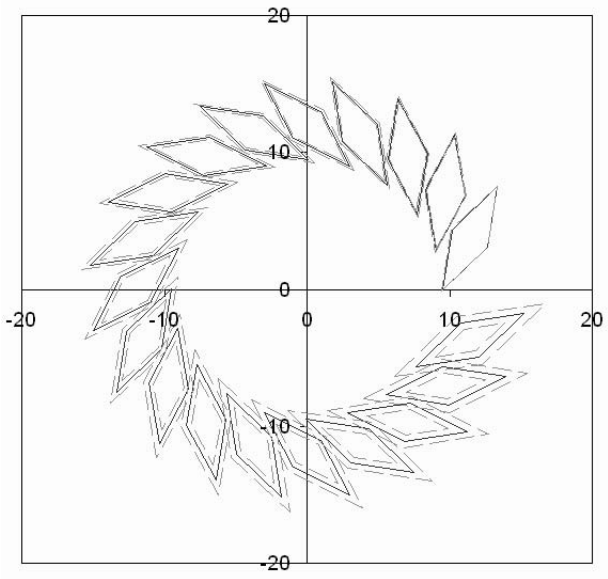


Figure 1. Inner and outer approximations of a linear map.

The second, denoted by $\mathbf{J}'$, is obtained by adding $[-0.001, 0.001]$ to each entry of $\mathbf{J}$. The characteristic matrix M_1 is chosen as proposed in the previous subsection, i.e., $M_1 = \mathrm{mid}\,(\mathbf{J}M_0)$. The initial parallelepipeds $\mathbb{A}_0$ and $\mathbb{B}_0$ are equal to $\{Mx | x \in \mathbf{x}\}$ with

$$M := \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \text{ and } \mathbf{x} := \begin{pmatrix} [7, 9] \\ [-5, -3] \end{pmatrix}. \qquad (33)$$

The parallelepiped approximations for the first 18 steps are displayed on the left hand side of Figure 1. Solid parallelepipeds are computed using $\mathbf{J}$. Inner and outer approximations are too close to be distinguishable. Dotted parallelepipeds are computed using $\mathbf{J}'$. This time, due to the poor quality of the LIM, inner and outer approximations quickly separate.

After 10^7 iterations, i.e., after 5×10^5 complete rotations, the inner and outer parallelepiped approximations computed using $\mathbf{J}$ are still less than 10^{-6} apart.

4 Inner and Outer Approximations of the Solutions to Uncertain Initial Value Problems

The initial value problem (IVP) consists of computing an approximation of the solution to some ordinary differential equation (ODE) for some initial value. The solution at time t can can be expressed naturally as a function of the initial value, function called the ODE *solution operator* (cf. subsection 4.1). So when one deals with a set of initial conditions, one needs to approximate the image of a set by the ODE solution operator. The results stated in the previous section can then be used, leading in particular to rigorous inner approximations. Up to our knowledge, the only work that proposes to compute such inner approximations is [20]. However, inner approximation is just mentioned in [20] and no detail can be found on the way such inner approximations can be computed.

4.1 General Framework

We follow the definitions of interval IVPs set out in [25]. First, given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and a vector $y_0 \in \mathbb{R}^n$, the IVP consists of computing a function y that satisfies

$$y'(t) = f\big(y(t)\big), \qquad (34)$$
$$y(0) = y_0 \in \mathbb{R}^n. \qquad (35)$$

The function f is assumed to be $N \geq 1$ times continuously differentiable. In practice, one wishes to compute an approximation of $y(h)$ for a given time step $h > 0$.

Remark 3. Here, we do not consider explicitly the construction of a solution step-after-step but instead focus on one single step. The composition of several steps can be done applying the method described at the end of Subsection 3.2.

Equation (34) is called the *defining equation*, and (35) the *initial value*. To simplify the presentation, we assume existence and uniqueness of the solution to (34) for all initial conditions. It is convenient to describe the solution of the IVPs with respect to the initial condition using a function $\phi_h : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ such that $y(t + h) = \phi_h(y(t))$. This mapping is well defined in $\mathbb{R}^n$ because we assumed existence and uniqueness of the solution to (34). The function ϕ_h is called the *time-h solution operator* of the ODE (34). Thanks to the definition of the ODE solution operator, the IVP problem can be cast into the problem that consists of computing some approximation of $\phi_h(y_0)$.

We now consider a set of initial conditions $\mathbb{Y}_0$, called an uncertain initial condition, and (35) is replaced by

$$y(0) \in \mathbb{Y}_0 \subseteq \mathbb{R}^n. \qquad (36)$$

We aim to construct some approximation of

$$\mathbb{Y}_h := \{y(h) \mid y \text{ satisfies (34) and (36)}\} \qquad (37)$$
$$= \mathrm{range}\big(\phi_h, \mathbb{Y}_0\big). \qquad (38)$$

We call this problem an uncertain initial value problem (UIVP). In practice, we suppose that $\mathbb{Y}_0$ is approximated by an inner and an outer parallelepiped, i.e., $\mathbb{A}_0 \subseteq \mathbb{Y}_0 \subseteq \mathbb{B}_0$ where $\mathbb{A}_0 := \{Uu \mid u \in \mathbf{a}_0\}$ and $\mathbb{B}_0 := \{Uu \mid u \in \mathbf{b}_0\}$. Then, we aim to construct two parallelepipeds $\mathbb{A}_h := \{Vv \mid v \in \mathbf{a}_h\}$ and $\mathbb{B}_h := \{Vv \mid v \in \mathbf{b}_h\}$ such that $\mathbb{A}_h \subseteq \mathbb{Y}_h \subseteq \mathbb{B}_h$. Such approximations can be computed by applying Theorem 2 in order to construct inner and outer approximations of $\mathrm{range}\big(\phi_h, \mathbb{Y}_0\big)$. Therefore, all we need are

(i) an outer approximation $\mathbf{z}$ of $\phi_h(y_0)$, and

(ii) a LIM of ϕ_h and $\mathbb{Y}_0$.

To compute (i), one can use the usual interval methods for computing enclosing approximation of IVP. To obtain (ii), an enclosure of the derivatives of the solution operator can be computed. Its computation can be expressed as a n^2+n components IVP. Therefore, the same interval methods can also be used to compute (ii). However, n^2+n quickly becomes too big for a direct use of interval IVP solvers (which use the derivatives of the ODE to be integrated, hence leading to $O(n^4)$ Taylor coefficients to be evaluated in this case). The next subsection provides direct methods to compute a LIM for the ODE solution operator.

4.2 Lipschitz Interval Matrices for the ODE Solution Operator

This subsection presents some techniques to compute a LIM for ϕ_h and $\mathbb{Y}_0$. Throughout the subsection, we consider two boxes $\mathbf{y}_0$ and $\mathbf{y}_{[0,h]}$ that contains respectively $\mathbb{Y}_0$ and $\mathbb{Y}_{[0,t]}$, i.e., $\mathbf{y}_0$ contains all initial conditions and $\mathbf{y}_{[0,h]}$ contains all trajectories for $t \in [0,h]$ and for any initial condition in $\mathbf{y}_0$. First, $\mathbf{y}_0 := M_0\mathbf{b}_0$ is the optimal enclosure of $\mathbb{Y}_0$ and is easy to compute. The computation of $\mathbf{y}_{[0,h]}$ can be more problematic. For small enough time step h, $\mathbf{y}_{[0,h]}$ can be computed using a rigorous first order approximation: it is indeed sufficient that $\mathbf{y}_{[0,h]}$ satisfies $\mathbf{y}_0 + hf(\mathbf{y}_{[0,h]}) \subseteq \mathbf{y}_{[0,h]}$ so that the Picard-Lindelöf operator proves existence and uniqueness of the solution inside $\mathbf{y}_{[0,h]}$ for any initial value in $\mathbf{y}_0$ (see e.g. Section 5 in [25]). Higher order and more efficient methods can also be used to compute $\mathbf{y}_{[0,h]}$.

4.2.1 Lipschitz Interval Matrices for Linear ODE

We first consider the case of a linear ODE, i.e., $f(y) = Ay$ with $A \in \mathbb{R}^{n\times n}$. In this case $\phi_h(y_0) = e^{hA}y_0$. Therefore, a LIM for ϕ_h is easily obtained:

Theorem 3. *Any interval matrix $\mathbf{J} \in \mathbb{IR}^{n\times n}$ that contains e^{hA} is a LIM for ϕ_h and $\mathbb{R}^n$.*

Proof. Consider any $x, y \in \mathbb{R}^n$. The $\phi_h(y) - \phi_h(x) = e^{hA}y - e^{hA}x$ which is equal to $e^{hA}(y-x)$. Because $e^{hA} \in \mathbf{J}$ by hypothesis, we have $\phi_h(y) - \phi_h(x) \in \mathbf{J}(y-x)$. □

Remark 4. An interval matrix $\mathbf{J}$ that contains e^{hA} can be constructed using truncated Taylor expansions with Householder norms for an accurate computation of the remainder [29], or by using Padé approximations [2].

Example 1. We consider the linear ODE defined by

$$y'(t) = Ay(t) \text{ with } A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{39}$$

Following the example presented in Subsection 3.4, we fix $h = \frac{\pi}{10}$. Using some rigorous interval approximation of $e^{\frac{\pi}{10}A}$ we obtain a LIM very close to (32). Using an initial condition $y(0) \in \{Mx | x \in \mathbf{x}\}$ with (33), the parallelepiped approximations are also similar to the one plotted in the left hand side of Figure 1.

4.2.2 Lipschitz Interval Matrices for Nonlinear ODE

In this section, some enclosures of the derivative of the solution operator w.r.t. the initial condition are computed instead of some LIM (the solution operator is actually differentiable w.r.t. the initial condition, cf. Theorem 14.3 in [10]). Let us recall that $\frac{\partial \phi_h(x)}{\partial x}\big|_y$ and $\frac{\partial f(x)}{\partial x}\big|_y$ are denoted by $\phi'_h(y)$ and $f'(y)$ respectively (no confusion will be possible because $\phi_h(y)$ will not be differentiated w.r.t. time). The following theorem can be found e.g. in [3]. It will provide us with a first enclosure of the derivative of the solution operator.

Theorem 4. *If $y(t)$ and $z(t)$ each satisfy the differential equation $y'(t) = f(y(t))$ on $[t_0, t_1]$, and f is Lipschitz continuous with constant L. Then $\forall t \in [t_0, t_1]$,*

$$||y(t) - z(t)|| \leq ||y(t_0) - z(t_0)||e^{L(t-t_0)}. \tag{40}$$

Remark 5. As both $y(t) \in \mathbf{y}_{[0,h]}$ and $z(t) \in \mathbf{y}_{[0,h]}$ for any $t \in [0,h]$, a Lipschitz constant for the restriction of f to $\mathbf{y}_{[0,h]}$ can be used in Theorem 4. As a consequence, one can use $L = ||\mathbf{f}'(\mathbf{y}_{[0,h]})||$.

The next corollary of Theorem 4 provides a crude enclosure of the derivative of the solution operator. It will not be used in practice, but it will be the basis to construct other more accurate enclosures.

Corollary 2. *Suppose that f is Lipschitz continuous in $\mathbf{y}_{[0,t]}$ with constant L, and define the interval matrix $\mathbf{J}_{[0,t]}$ by $(\mathbf{J}_{[0,t]})_{ij} := \left[-e^{hL}, e^{hL}\right]$. Then, $\mathbf{J}_{[0,t]} \supseteq \{\phi'_t(y_0) \mid y_0 \in \mathbf{y}_0\}$ for all $t \in [0,h]$.*

Sketch of the proof. By definition, $\phi'_t(y_0)$ satisfies

$$\phi_t(y_0 + \epsilon\delta) - \phi_t(y_0) = \epsilon\phi'_t(y_0) \cdot \delta + O(\epsilon^2), \tag{41}$$

where $\delta \in \mathbb{R}^n$ and $0 < \epsilon \in \mathbb{R}$. Choose $\delta = e_j$, where e_j is the j^{th} base vector and note that $(\phi'_t(y_0)\cdot e_j)_i = \big((\phi'_t(y_0)\big)_{ij}$. One obtains

$$\big(\phi_t(y_0 + \epsilon e_j)\big)_i - \big(\phi_t(y_0)\big)_i = \epsilon\big(\phi'_t(y_0)\big)_{ij} + O(\epsilon^2). \tag{42}$$

As a consequence,

$$|\big(\phi_t(y_0 + \epsilon e_j)\big)_i - \big(\phi_t(y_0)\big)_i| = \epsilon|\big(\phi'_t(y_0)\big)_{ij}| + O(\epsilon^2), \tag{43}$$

which contradicts Theorem 4 if $|\big(\phi'_t(y_0)\big)_{ij}| > e^{tL}$. □

To obtain a better enclosure of the derivatives of ϕ, Taylor expansions of ϕ_h and ϕ'_h w.r.t. time are now considered. To this end, the auxiliary functions $y^{[k]}(x)$ are introduced: as f is N times continuously differentiable, $y(t)$ is $N+1$ times continuously differentiable and $y^{(k)}(t)$, for $k \in \{1,\ldots,N+1\}$, can be computed as a function of $y(t)$. For example in dimension one,

$$y'(t) = f\big(y(t)\big), \tag{44}$$
$$y''(t) = f'\big(y(t)\big)f\big(y(t)\big), \tag{45}$$
$$y'''(t) = f''(y(t))f(y(t))^2 + f'(y(t))^2 f(y(t)), \tag{46}$$

etc. In order to easily manipulate these expressions, we define $y^{[k]}(x)$ such that

$$y^{[0]}\big(y(t)\big) = y(t), \tag{47}$$
$$y^{[k]}\big(y(t)\big) = y^{(k)}(t). \tag{48}$$

So, identifying (48) with equations (44–46), we obtain

$$y^{[1]}(x) = f(x), \tag{49}$$
$$y^{[2]}(x) = f'(x)f(x), \tag{50}$$
$$y^{[3]}(x) = f''(x)f(x)^2 + f'(x)^2 f(x). \tag{51}$$

As f is assumed N times continuously differentiable, $y^{[k]}$ is well-defined and continuously differentiable for $k \in \{1,\ldots,N\}$. The general recursive expressions for $y^{[k]}$ are easily obtained:

$$y^{[1]}(x) = f(x), \tag{52}$$
$$y^{[i]}(x) = \frac{\partial y^{[i-1]}(x)}{\partial x} \cdot f(x). \tag{53}$$

The functions $y^{[k]}$ can be evaluated either by computing formally their expressions or using automatic differentiation (see e.g. [31, 25]).

Remark 6. The functions $f^{[k]}(x) := \frac{1}{k!}y^{[k]}(x)$ instead of $y^{[k]}(x)$ are defined in [25]. Including $\frac{1}{k!}$ in the expressions is important in implementations as it usually allows stabilizing the recursive evaluation of the expressions while slightly reducing the computational cost of the evaluation.

Then, we obtain an expression of $\phi_h(y_0)$ w.r.t. y_0 using Taylor's theorem:

$$\phi_h(y_0) = \sum_{k=0}^{N-1} \frac{h^k}{k!} y^{[k]}(y_0) + r(y_0), \tag{54}$$

where we use Cauchy's remainder

$$r(y_0) := \int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\, y^{[N]}\big(\phi_t(y_0)\big)\, dt, \tag{55}$$

the integration being understood componentwise. The next theorem provides a way to improve the first enclosure computed, thanks to Corollary 2.

Theorem 5. *Let $\mathbf{y}_{[0,h]}$ and $\mathbf{J}_{[0,h]}$ be such that*

$$\mathbf{y}_{[0,h]} \supseteq \{\phi_t(y_0) \mid t \in [0,h], y_0 \in \mathbf{y}_0\} \tag{56}$$
$$\mathbf{J}_{[0,h]} \supseteq \{\phi'_t(y_0) \mid t \in [0,h], y_0 \in \mathbf{y}_0\}. \tag{57}$$

Then,

$$\sum_{k=0}^{N-1} \frac{h^k}{k!} \left.\frac{\partial y^{[k]}(x)}{\partial x}\right|_{\mathbf{y}_0} + \frac{h^N}{N!} \left.\frac{\partial y^{[N]}(x)}{\partial x}\right|_{\mathbf{y}_{[0,h]}} \cdot \mathbf{J}_{[0,h]} \tag{58}$$

is an enclosure of $\{\phi'_h(x) \mid x \in \mathbf{y}_0\}$.

Proof. Differentiating (54) w.r.t. y_0 gives rise to the following expression for $\phi'_h(y_0)$:

$$\sum_{k=0}^{N-1} \frac{h^k}{k!} \frac{\partial y^{[k]}(y_0)}{\partial y_0} + \frac{\partial r(y_0)}{\partial y_0}. \tag{59}$$

Now, let us explicit the expression of $\frac{\partial r(y_0)}{\partial y_0}$ differentiating inside the integration and using the chain rule: $\frac{\partial r(y_0)}{\partial y_0} =$

$$\int_0^h \frac{(h-t)^{N-1}}{(N-1)!} \left.\frac{\partial y^{[N]}(x)}{\partial x}\right|_{\phi_t(y_0)} \cdot \left.\frac{\partial \phi_t(x)}{\partial x}\right|_{y_0} dt. \tag{60}$$

Now, using (56) and (57),

$$\underline{A} \le \left.\frac{\partial y^{[N]}(x)}{\partial x}\right|_{\phi_t(y_0)} \cdot \left.\frac{\partial \phi_t(x)}{\partial x}\right|_{y_0} \le \overline{A}, \tag{61}$$

where

$$[\underline{A}, \overline{A}] := \left.\frac{\partial y^{[N]}(x)}{\partial x}\right|_{\mathbf{y}_{[0,t]}} \cdot \mathbf{J}_{[0,h]} \in \mathbb{IR}^{n\times n}. \tag{62}$$

From (60) and (61), and because $h - t \ge 0$ holds for all $t \in [0,h]$, the following inequalities hold:

$$\int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\underline{A}\, dt \le \frac{\partial r(y_0)}{\partial y_0} \le \int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\overline{A}\, dt, \tag{63}$$

and hence obviously

$$\underline{A}\int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\, dt \le \frac{\partial r(y_0)}{\partial y_0} \le \overline{A}\int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\, dt. \tag{64}$$

Finally, noting that

$$\int_0^h \frac{(h-t)^{N-1}}{(N-1)!}\, dt = \frac{h^N}{N!}, \tag{65}$$

one concludes the proof. □

Remark 7. As for evaluating $y^{[k]}$, evaluating $\frac{\partial y^{[k]}(x)}{\partial x}$ can be done computing its formal expression or using automatic differentiation (see e.g. [31, 25]).

Remark 8. The interval matrix $\mathbf{J}_{[0,h]}$ needed in Theorem 5 is computed using Corollary 2. It is useful to improve this first crude enclosure by

$$\mathbf{J}_{[0,h]} \leftarrow \Big(\sum_{k=0}^{N-1} \frac{[0,h]^k}{k!} \cdot \frac{\partial y^{[k]}(x)}{\partial x}\Big|_{\mathbf{y}_0} \tag{66}$$

$$+ \quad \frac{[0,h]^N}{N!} \cdot \frac{\partial y^{[N]}(x)}{\partial x}\Big|_{\mathbf{y}_{[0,h]}} \cdot \mathbf{J}_{[0,h]} \Big) \tag{67}$$

$$\bigcap \quad \mathbf{J}_{[0,h]}, \tag{68}$$

which obviously maintains $\mathbf{J}_{[0,t]} \supseteq \{\phi'_t(y_0) \mid y_0 \in \mathbf{y}_0\}$ for all $t \in [0,h]$ due to Theorem 5. Repeated application can improve the enclosure.

In the special case of first order (58) is

$$I \; + \; hf'(\mathbf{y}_{[0,h]}) \cdot \mathbf{J}_{[0,h]}. \tag{69}$$

For small enough values of h, this simple expression together with the improvement process of Remark 8 can already provide useful enclosures of $\{\phi'_h(x) \mid x \in \mathbf{y}_0\}$. Let us first illustrate Theorem 5 with the special case of a linear ODE.

Example 2. Let $f(y) = A \cdot y$. In this case, we have $y(t) = e^{tA} \cdot y_0$, $y'(t) = A \cdot y(t)$, $y''(t) = A^2 \cdot y(t)$ and so on. Finally, one proves that $y^{(k)}(t) = A^k \cdot y(t)$ and therefore $y^{[k]}(x) = A^k \cdot x$. As a consequence,

$$\frac{\partial y^{[k]}}{\partial x}(x) = A^k. \tag{70}$$

Formula (58) therefore gives rise to

$$I \; + \; \sum_{k=1}^{N-1} \frac{h^k}{k!} A^k \; + \; \frac{h^N}{N!} A^N \cdot \mathbf{J}_{[0,h]}, \tag{71}$$

where Corollary 2 allows $(\mathbf{J})_{ij} = [-e^{h||A||}, e^{h||A||}]$. This can be interpreted as a rigorous truncation of the series e^{hA}.

Example 3. The Lorenz system is defined by

$$\begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} \sigma\big(y(t) - x(t)\big) \\ x(t)\big(\rho - z(t)\big) - y(t) \\ x(t)y(t) - \beta z(t) \end{pmatrix}. \tag{72}$$

We use the usual values $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$, for which the system exhibits chaotic behavior [21]. The uncertain initial condition is chosen to be $\mathbf{y}_0 = \big(10 \pm 10^{-10}, 10 \pm 10^{-10}, 10 \pm 10^{-10}\big)$. Then applying Theorem 5 with 20^{th} order expansions and a timestep of $h = 0.02$ gives rise to one-timestep distance between the inner and outer approximations of less than 4×10^{-14}.

4.2.3 Related Work

Only a very few references deal with the problem of a rigorous computation of a LIM for an ODE solution operator. The parallelepiped method proposed by Kruckeberg in [20] implicitly uses

$$\sum_{k=0}^{\infty} \frac{h^k \mathbf{J}_f^k}{k!}, \tag{73}$$

where $\mathbf{J}_f \supseteq \{f'(x) \mid x \in \mathbf{y}_0\}$, as a LIM for ϕ_h. However the proof of this property is not detailed in [20] (see formula 29 page 95[3]). And more important, no rigorous truncation of (73) is available. The rigorous truncation proposed in [29] cannot be applied to (73) because $\{\exp(hJ_f) \mid J_f \in \mathbf{J}_f\}$ is not equal to (73), while no simplification (e.g. Horner scheme) can be applied to (73) until one has proved the simplified formula is also a LIM for ϕ_h (i.e., it is likely to happen that the overestimation of $\{\exp(hJ_f) \mid J_f \in \mathbf{J}_f\}$ in the expression (73) is necessary to obtain a LIM).

Stauning [31] proposes (formula 6.17 and 6.18 page 69) the following LIM:

$$I \; + \; \sum_{k=1}^{N-1} \frac{1}{k!} \frac{\partial y^{[k]}}{\partial y}(\mathbf{y}_0) h^k \; + \; \frac{1}{N!} \frac{\partial y^{[N]}}{\partial y}(\mathbf{y}_{[0,h]}) h^N \tag{74}$$

However, (74) is not a LIM for ϕ_h and $\mathbf{y}_0$ in general (in the case of a linear system, (74) just truncates the series e^{tA} without providing any remainder). The correct formula is the one provided in Theorem 5.

Makino [22] notes that Taylor models cannot help computing a LIM for ϕ_h and just mentions a potential method for the computation of $\mathbf{J}$ (cf. page 92 of [22]).

Finally, the authors have recently discovered the work of Zgliczynski [32] where the derivative of the solution operator is rigorously enclosed. Though both works are similar, our presentation seems simpler while providing some similar enclosures. The exact relationship between the two methods remains to be investigated.

5 Conclusion

While the outer approximation of the range of a function is a basic application of interval analysis, the inner approximation of the range remains today a problem not well solved. A new procedure for the computation of such an inner approximation has been proposed, based on a corollary of the Poincaré-Miranda theorem: for roughly no additional cost, one is now able to compute an inner approximation together with the outer approximation given by the mean-value extension. The inner approximation will not be

[3] A scanned version of [20] is available at http://www.goldsztejn.com/downloads.htm

empty only in the situations where the mean-value extension provides a sharp enclosure. In particular, the interval Lipschitz matrix used has to be an H-matrix. A specific preconditioning process has been proposed to help fulfilling this necessary condition, leading to parallelepiped approximations. Due to some potential applications in the theory of shadowing dynamical systems, some properties that allow us to compute Lipschitz interval matrices in the context of ordinary differential equations have been presented.

Experiments are currently underway to apply these developments to the rigorous shadowing of dynamical systems (cf. [13, 14]), using ideas similar to those proposed in Section 3.5.4 of [12].

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Computer Science and Applied Mathematics, 1974.

[2] P. Bochev and S. Markov. A Self-Validating Numerical Method for the Matrix Exponential. *Computing*, 43(1):59–72, 1989.

[3] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations*. Wiley, New York, 1987.

[4] A. Frommer, B. Lang, and M. Schnurr. A Comparison of the Moore and Miranda Existence Tests. *Computing*, 72:349–354, 2004.

[5] A. Goldsztejn. Verified Projection of the Solution Set of Parametric Real Systems. In *Proc. of COCOS 2003*.

[6] A. Goldsztejn. *Définition et Applications des Extensions des Fonctions Réelles aux Intervalles Généralisés*. PhD thesis, Université de Nice-Sophia Antipolis, 2005.

[7] A. Goldsztejn. A Comparison of the Hansen-Sengupta and Frommer-Lang-Schnurr Existence Theorems. *Computing*, 79(1):53–60, 2007.

[8] A. Goldsztejn, D. Daney, M. Rueher, and P. Taillibert. Modal Intervals Revisited: a Mean-Value Extension to Generalized Intervals. In *Proc. of QCP 2005*.

[9] A. Goldsztejn and L. Jaulin. Inner Approximation of the Range of Vector-Valued Functions. *Submitted to Reliable Computing*.

[10] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Springer-Verlag, 2000.

[11] B. Hayes. A Lucid Interval. *American Scientist*, 91(6):484–488, 2003.

[12] W. Hayes. *Rigorous Shadowing of Numerical Solutions of Ordinary Differential Equations by Containment*. PhD thesis, University of Toronto, 2001.

[13] W. Hayes and K. R. Jackson. Rigorous Shadowing of Numerical Solutions of Ordinary Differential Equations by Containment. *SIAM J. Numer. Anal.*, 41(5):1948–973, 2003.

[14] W. Hayes and K. R. Jackson. A Survey of Shadowing Methods for Numerical Solutions of Ordinary Differential Equations. *Applied Numerical Mathematics*, 53(1-2):299–321, 2005.

[15] P. Herrero, M. Sainz, J. Vehí, and L. Jaulin. Quantified Set Inversion Algorithm with Applications to Control. In *Proc. of Interval Mathematics and Constrained Propagation Methods*, volume 11(5) of *Reliab. Comp.*, 2005.

[16] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, 2001.

[17] R. B. Kearfott. Interval Arithmetic Techniques in the Computational Solution of Nonlinear Systems of Equations: Introduction, Examples, and Comparisons. *Computational Solution of Nonlinear Systems of Equations, Amer. Math. Soc.*, 26:337–358, 1990.

[18] R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath*, Bulletin 2(1):95–112, 1996.

[19] R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary, and P. Van Hentenryck. Standardized Notation in Interval Analysis. 2002.

[20] F. Kruckeberg. Ordinary Differential Equations. In E. Hansen, editor, *Topics in Interval Analysis*, pages 91–97. Oxford University Press, 1969.

[21] E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[22] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, 1998.

[23] R. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[24] R. Moore and J. Kioustelidis. A Simple Test for Accuracy of Approximate Solutions to Nonlinear (or Linear) Systems. *SIAM J. Numer. Anal.*, 17(4):521–529, 1980.

[25] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated Solutions of Initial Value Problems for Ordinary Differential Equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[26] S. N. Nedialkov and K. R. Jackson. A New Perspective on the Wrapping Effect in Interval Methods for Initial Value Problems for Ordinary Differential Equations. In *Perspectives on Enclosure Methods*, pages 219–264. Springer-Verlag, 2001.

[27] V. M. Nesterov. Interval and Twin Arithmetic. *Reliab. Comp.*, 3(4):369–380, 1997.

[28] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, Cambridge, 1990.

[29] E. Oppenheimer and A. Michel. Application of Interval Analysis Techniques to Linear Systems. II. The Interval Matrix Exponential Function. *IEEE Transactions on Circuits and Systems*, 35(10):1230–1242, 1988.

[30] H. Poincaré. Sur Certaines Solutions Particulières du Problème des Trois Corps. *Comptes rendus de l'Académie des sciences*, 97:251–252, 1883.

[31] O. Stauning. *Automatic Validation of Numerical Solutions*. PhD thesis, Technical University of Denmark, 1997.

[32] P. Zgliczynski. C^1-Lohner Algorithm. *Foundations of Computational Mathematics*, 2(4):429–465, 2002.

Ensuring Numerical Quality in Grid Computing

Andreas Frommer, Matthias Hüsken
Bergische Universität Wuppertal
Applied Computer Science and Scientific Computing
D-42097 Wuppertal, Germany
frommer@math.uni-wuppertal.de, huesken@math.uni-wuppertal.de

Abstract

Certain numerically intensive applications executed within a grid computing environment crucially depend on the properties of floating-point arithmetic implemented on the respective platform. Differences in these properties may have drastic effects. This paper identifies the central problems related to this situation. We propose an approach which gives the user valuable information on the various platforms available in a grid computing environment in order to assess the numerical quality of an algorithm run on each of these platforms. In this manner, the user will at least have very strong hints whether a program will perform reliably in a grid before actually executing it.

Our approach extends the existing IeeeCC754 test suite by two "grid-enabled" modes: The first mode calculates a "numerical checksum" on a specific grid host and executes the job only if the checksum is identical to a locally generated one. The second mode provides the user with information on the reliability and IEEE 754-conformity of the underlying floating-point implementation of various platforms. Furthermore, it can help to find a set of compiler options to optimize the application's performance while retaining numerical stability.

1. Introduction

Although the numerical problems caused by representation issues of real numbers on a computer have always been an important issue, it has taken a long time to become common knowledge that software developers must be aware of the pitfalls of floating-point arithmetic and number conversion and that they have to account for the behavior of the floating-point arithmetic when developing software. Fortunately, disasters due to problems with conversion and arithmetic like the two listed in [1] (a failure of the patriot anti-missile system and the explosion of an Ariane 5 rocket) have become seldom, but reliability of the underlying floating-point implementation is still an issue of highest importance. This is especially true when thinking of reliable computing and interval arithmetic as all practical implementations assume the underlying floating-point implementation to be error-free and standards-compliant.

Today, every major hardware platform and almost all available software development tools (like compilers and mathematical libraries) implement floating-point arithmetic to be IEEE 754 compliant. Unfortunately, results generated on different platforms still can not be assumed to be totally comparable due to the fact that no-one can guarantee that a given floating-point implementation is indeed correct (i. e., error-free). But what starts making things really involved is that, usually, different operations specified in IEEE 754 might be implemented either in hardware or software (for details see Section 2). So when running an application on different platforms, parts of the floating-point computations might be handled by completely different processors or software libraries (or at least different versions of the same library). Finally, to further complicate the situation, some compiler options can completely eliminate IEEE compliance,[1] a fact many users are still unaware of. So, *numerical reliability* is still an important issue today.

The picture outlined so far is of particular importance in *distributed computing*, where, anyway, compatibility issues still are one of the most challenging subjects. The concept of Grid computing [5] offers an (at least partially) standardized approach to distributing applications in a large-scale environment. But this effort does not address the problem of numerical reliability raised above.

In the following sections, we will develop approaches to ensure (or, at least, to increase the probability) that results generated by executing an application on different grid nodes will be identical and, ideally, be conforming to the IEEE standard. We view our developments as a useful tool

[1] This mainly applies to options affecting exception handling like `-ffast-math` or `-fno-trapping-math` in g++ or options that enable higher optimization settings as these affect those parts of the floating-point implementation that are internally generated by the compiler.

for application programmers who are aware of the particular behavior of floating-point arithmetic and who have developed software assuming that floating-point arithmetic behaves predictably and, preferably, according to the IEEE standard. In Section 3 we propose a method of checking a grid node for numerical compatibility and preventing a submitted job from running when a corruption of results is likely to occur. Section 4 deals with the use of the information systems already present in grid toolkits to generate information on the numerical reliability of the grid nodes and to store it (and to make it available) in these systems. Special emphasis will be put on the influence of compiler options. Section 5 presents a side use of one of the approaches in Section 4 that can be used to enhance the performance of a given application while still retaining numerical quality. Finally, we present some test results within the research grid at our department and the LCG (LHC Computing Grid) and conclude with a summary.

2. IeeeCC754

Since publication of the IEEE standards 754 [7] and 854 [8], quite a number of tools for testing floating-point implementations have been developed. The grid tools presented in this paper are extensions to one of the latest test packages which is also one of the most comprehensive so far: IeeeCC754, which has been and is still further developed by the Computer Arithmetic and Numerical Techniques group at the University of Antwerp under leadership by A. Cuyt and B. Verdonk (cf. [15] and [16]). IeeeCC754 is based on ideas and principles of some other packages (a detailed description can be found in [15] and [16]) by combining and extending most of their properties, enabling it to test all operations defined in IEEE 754 and 854 at arbitrary precisions in all four rounding modes (provided that the floating-point implementation supports these) and detect the underflow mode used in the implementation.

IeeeCC754 itself is implemented in C++ and has been used to test a variety of hardware platforms (Intel compatible, SUN Sparc, embedded processors, proprietary chips, ...) in combination with several compilers (GNU g++, Intel icpc, SUN CC, SUN f95, ...). It has also been used to test a number of multiprecision libraries for compatibility with the principles of the IEEE 754 and 854 floating-point standards.

The package is divided into two parts: A driver program and a suite of precision- and range-independent test vectors. The main idea is the following: The driver program takes a test vector, converts the operands and the (correct) result into the binary floating-point format (with the desired precision), executes the specified operation and compares the computed result to the expected value. Furthermore, for each operation it is checked whether the implementation returned the exception flags required by the standard. If something is found to be non-compliant, an error message is printed into a log file, and the driver program continues with the next test vector. When all test vectors have been processed, a summary is generated and it is checked which underflow mode is implemented.[2] Figure 1 illustrates the architecture of IeeeCC754.

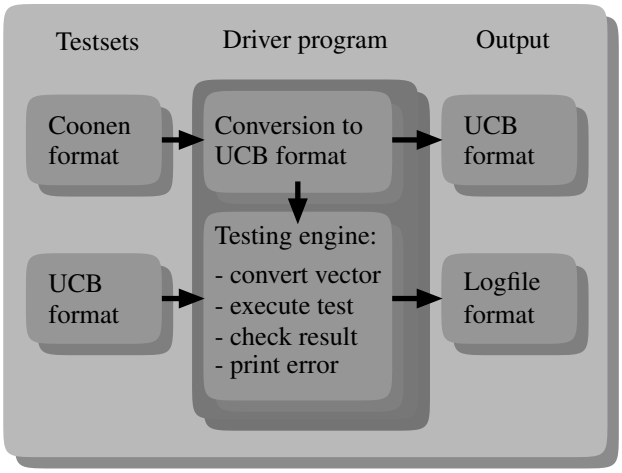


Figure 1. Architecture of IeeeCC754

IeeeCC754 uses two test vector formats which employ "extended Coonen" and "extended UCBTEST" syntax. An exact description in Backus-Naur form (BNF) as well as a short history of both formats can be found in [15] and [16].

The driver program supports three modes: The first one takes input in extended Coonen syntax and executes the testing stage, the second one takes test vectors in extended UCBTEST syntax as input and runs the tests, and finally there is a conversion mode to translate test vectors from extended Coonen syntax to extended UCBTEST syntax. IeeeCC754 features comprehensive testsets in extended Coonen format for all operations listed in IEEE 754.

IeeeCC754 distinguishes between basic operations and conversions. This distinction is motivated by the observation that on most systems certain operations like addition, multiplication and division are realized in (and mapped to) hardware functions, while code for other operations will be generated by the compiler and executed in software, e. g. conversions between decimal and binary format or between different floating-point formats. The basic operations as used in IeeeCC754 consist of $+$, $-$, $*$, $/$, square root and remainder. On platforms with SUN, Intel and AMD processors, the hardware provides implementations of these oper-

[2] Obviously, the underflow mode can only be determined if floating-point operations that produce underflow have been executed. This is noteworthy as certain kinds of operations never will result in underflow (cf. [2]).

ations, and testing usually reveals no errors. As for conversions, the code generated is highly dependent on the compiler and the runtime libraries to which the program will be dynamically linked to. Testing results vary from platform to platform, with the most errors usually found in decimal to binary (and vice versa) conversions as these are the functions that are the most complex to implement and most expensive to execute. For further details we refer to [16].

Summarizing, IeeeCC754 is a valuable tool to test to what degree a given floating-point implementation conforms to the IEEE 754 standard. The new modes presented in the next sections have been integrated into the IeeeCC754 development tree, due to the kind permission of the University of Antwerp to modify and extend their sources.

3. IeeeCC754 checksum mode

3.1. General approach

Our first approach to providing numerical quality in a grid environment is motivated by the heterogeneous nature of the platforms encountered in a grid. Even if a specific architecture (like "i386 running Linux") is requested when submitting a job, one cannot know in advance the exact specifications of the target host (like exact processor type, kernel version etc.). So, it is almost impossible to predict the job's results as far as floating-point behavior is concerned. To make things worse, most probably the platform on which the user locally develops his or her application will not totally conform to IEEE 754 in all aspects. Furthermore, even when the compiler is in principle able to generate IEEE compliant numerical code, this does not have to be the case when using certain optimization levels to gain higher application performance. Typically, compilers tend to drop certain aspects of IEEE 754 at higher optimization levels (such as generating floating-point exceptions) due to their computational cost. The influence of compiler options on floating-point arithmetic is completely dependent on the specific compiler and might even change from one compiler release to the next. Some examples of applications suffering loss of IEEE 754 compliance due to compiler options can e. g. be found in [12] or [4].

What can be done to get reliable and predictable results under these circumstances? The only facts the user knows about his or her application's performance is how it performs locally, i. e. when run on his or her local machine. Information about the degree of compliance to IEEE 754 of the local platform can be obtained by running standard mode IeeeCC754, even when running higher optimization levels (cf. Section 5). In this manner, some control on the local performance of the application can be achieved. Our approach to transfer this knowledge to the grid starts by calculating a "numerical checksum" of the local system, incorporating the set of compiler options that were used to generate the final application. When submitting the application to the grid, prior to actually executing it, the numerical checksum of the target grid host is computed and compared to the local checksum. If the checksums differ, it is likely that the results generated on that system will be different to those that would have been calculated when running the application on the local system. So the user will not be able to rely on the results when executing jobs on this specific grid host and, consequently, the grid job should be rejected and its execution canceled.

On the other hand, it is clear that even when getting identical checksums it can never be guaranteed that the application will indeed yield exactly the same results when run on a (positively) tested grid host or when run locally. But since the IeeeCC754 test suite is fairly comprehensive, the user at least gains a very strong hint that most likely the application will behave as expected.

As an important note we would like to stress that this approach will not (and cannot) provide the user with any information on the IEEE compliance of the underlying floating-point implementation(s) even if the checksums are equal. It simply provides a security means enabling the user to cancel (or to not start) jobs that are likely to perform unsatisfactorily. When information on IEEE 754 compliance is desired, standard mode IeeeCC754 should be used.

3.2. Implementation details

To calculate the checksum, we extended IeeeCC754 with an additional mode, the checksum mode. In this mode, IeeeCC754 reads test vectors in extended UCBTEST format (which means the test vectors have to be converted first, see below). During the testing stage, instead of producing error log files in plain text format, a binary format is used to keep the checksum compact.

The checksum consists of a header (21 bytes long) which includes the number of total vectors tested, the number of errors that occurred, the underflow mechanism used (and if it was used consistently), and finally the endianess of the platform on which the checksum was generated. Additionally, for every error that was encountered while executing IeeeCC754 (i. e., executing the operations under consideration led to a violation of IEEE 754), another 14 bytes will be written including information about error type etc. This (binary) encoding is done in such a way that it is possible to retrieve almost all the information from the checksum that would also have been produced with standard mode IeeeCC754 except the floating-point numbers (i. e. operands and expected and returned results) themselves. For the deployment of the checksum mode in a grid environment it might not be strictly necessary to retain this information, but it enables the user to actually locate the ar-

eas (i. e. operations) where differences occur by using the decoding program which was developed together with the checksum mode.

Due to the different philosophies behind the standard modes and the checksum mode,[3] we introduced a few small changes to the default behavior when extending IeeeCC754. First, standard IeeeCC754 has been designed to test the conformity of the single and double floating-point formats defined in IEEE 754. Intel-compatible processors execute floating-point operations in an extended format which uses extra bits for exponent and mantissa (at least by default). The operands (in single or double precision) are copied to the internal format, the operation is executed with extended precision and then rounded back to default precision. Although this yields results with higher accuracy (which is numerically desirable), it breaks IEEE 754 conformity (because computing with a longer mantissa may lead to different results when rounding). System calls exist to change the default behavior and switch the processor to IEEE single or double mode, but even then the exponent is given more bits and thus can vary in a bigger range. During testing IeeeCC754 sets the correct modes via these system calls. We disable the switching of floating-point modes when running on Intel-compatible hardware, since in most cases a "normal user" will simply use the default single and double datatypes provided by the programming language. Thus, more errors will be encountered when executing single and double precision tests. This is not a problem by itself (the goal of the checksum mode is to generate a security measure, not to provide conformity information), but it leads to a considerably larger checksum.

Another issue on platforms with Intel-compatible processors demands attention: Although the hardware implements functions for the computation of square root and remainder which yield (mostly) IEEE 754 compliant results, some compilers generate their own version of these operations. In particular, when using g++ (the GNU compiler) or icpc (the Intel compiler) on a Linux i386 platform, square root and remainder are realized in software. To test the quality of the processor's floating-point implementation, standard IeeeCC754 uses assembler calls for these operations. Again, we try to mimic the computing environment most users will see and rely on the operations which the compiler provides by default. Typically, this leads to some additional errors.

When using IeeeCC754 checksum mode, there are a few points worth considering. Obviously, a certain overhead will be produced by transferring IeeeCC754 itself, the testsets and the local checksum to the grid node. The size of the (probably already quite large, see above) checksum file will be further increased (and to a much larger extent) when using high optimization levels. Another issue (which is also related to the checksum file size) is the number of test vectors to be considered. It is possible to create rather small testsets by omitting certain operations or precisions. E.g., it is quite feasible to drop all vectors concerning long[4] precision on Intel-compatible platforms since most probably the native format will not be used (or may not even be accessible) by the user anyway. In the same manner, quadruple precision test vectors might be dropped on SUN platforms when the application makes no use of them. Furthermore, if only the default rounding mode is used (which is "round to nearest" as specified by IEEE 754), it makes sense to generate only vectors that test this mode. Although possible in theory, it is not advisable to omit the complete testset for a given operation since valuable (and necessary) information might be lost. In particular, it is dangerous to omit test vectors from the conversion test sets as certain conversions might be used by the compiler implicitly without the user ever being aware of it.

As a small testset generates less overhead in a grid and reduces the size of the checksum file at the same time, it is planned to offer a specially designed testset which contains only test vectors that are known to lead to errors with a high probability. Before this testset can be released, many more test results will be necessary to ensure that (if possible) almost no information is lost due to the reduction.

We propose another approach to drastically reduce the size of the checksum file: Header and error information are generated as described above, but instead of interpreting all this information as the checksum, we compute a hash value (MD5 [11] or SHA-1 [10]) of the encoded error information and add that value to the header. Header and hash value are now the new checksum. It is of fixed size (21 bytes plus the length of the hash value which is 128 bits for MD5 and 160 bits for SHA-1). This is the most desirable way to deploy the checksum mode in a grid (together with a reduced testset) because of its efficency – checksums can be as big as 140 KB (which corresponds to more than 10,000 errors!). Still, for analysis purposes the original long checksum should be used.

Two drawbacks related to this approach should be mentioned: When computing hash functions, there is inherently always a small probability that two different checksums might result in the same hash value. Although modern hashing functions (which are mainly used to ensure file integrity) are designed to avoid collisions especially for binary input data, a small risk remains that an application will be executed although the grid node is to be considered unreliable (due to different checksums).

[3] The standard modes test the principal ability of a floating point implementation to support IEEE 754, whereas the checksum mode checks the compatibility of different nodes from a "user-centric" programming environment point of view.

[4] "long" is the extended format that is used internally in Intel-compatible processors.

Furthermore, by computing the hash value of the checksum, it is no longer possible to retrieve information on the areas in which a given grid node differs from the local platform with regard to numerical reliability.

4. IeeeCC754 info mode

4.1. General approach

While the checksum mode aims at providing a means to prevent the execution of applications on grid nodes which are likely to produce unreliable results, we propose another approach that makes use of the information modules present in most grid toolkits. These systems publish different pieces of information about the nodes present in a grid. The information published is typically used by resource managers which try to find the best currently available resources for a job that has been submitted, but they also help users and system administrators to identify and solve problems typically encountered in a grid.

Information systems are present in every major grid toolkit, so it seems quite obvious to publish information of the numerical properties of a grid node into these systems. The question is which information about an underlying floating-point implementation (that cannot be derived from other already existing information like processor type) is useful enough to be worth being published into a grid? In the following, we explain the implementation of the IeeeCC754 info mode and propose three approaches that provide information that could be helpful when published on a grid.

The info mode actually consists of two parts – an extension to IeeeCC754 which in particular introduces an adequate output format, and a set of python scripts which control the generation of test results, collect statistics and produce output which can be published.

In many regards, the IeeeCC754 info mode is very similar to the checksum mode (see above). It expects testsets in extended UCBTEST format, and on Intel-compatible platforms it uses the same floating-point environment a "normal user" sees (i. e., it relies on the compiler-generated software versions for square root and remainder and does not switch to IEEE compliant single and double formats via system calls). The main difference is the output format used: As the goal of the info mode is to generate statistics, a line of output is generated for every test vector being considered (standard IeeeCC754 and check mode only produce output when an error has been detected). The output format is defined as follows:

- For successful tests:
  `+ prec op mode`
- For errors:
  `- prec op mode error`

where the first character shows whether testing resulted in a success (+) or in an error (−), `prec` denotes the precision of the result, `op` denotes the operation that has been tested, and `mode` denotes the rounding mode. Furthermore, in case an error was encountered, the type of error is recorded. Here we find the second big difference to the other IeeeCC754 modes: We classify all errors that can occur into two groups. One contains all errors that deal with exception flags, be it flags that have been returned but should not or exceptions that are expected but were not raised. The other group consists of results where at least one bit of the actually returned floating-point number is wrong, irrespective of the bit position (sign, exponent or mantissa). The motivation behind this distinction is simple: In a common development environment, exception flags are only available to the user when explicitly requested (usually via system calls). Then, the execution of an application is not affected by errors due to erroneously set exceptions flags at all. On the other hand, when at least one bit of the returned floating-point number is set to a wrong value, a non-compliant numerical error is introduced in the computation which may result in a loss of accuracy. With the classification approach, one can distinguish between severe and not so relevant errors. In info mode, an `x` is printed if an error in the exception class occurred, an `r` when the computed binary representation of the result is not correct, and finally `xr` when both types of errors occurred.

4.2. The generation framework

The "generation framework" is designed to be as flexible as possible to enable output of a wide range of information for arbitrary grid information systems. It includes functionality to execute the testing stage with a given set of compiler options as well as functionality to find the "best" set of compiler options according to different criteria (cf. Section 5). The functionality is shown in figure 2.

Selection of the relevant information is done by either choosing specific test vectors from the testsets or, in a more flexible manner, by filters: In the parsing stage, the framework reads the results from IeeeCC754 and stores them in a hierarchical data structure optimized for fast selection of certain criteria. Afterwards, a filter is applied to the data.

A filter consists of two components: A list of criteria that specifies which of the test results should be considered, and a filter function that is successively applied to every result that satisfies the given criteria. The filter function is responsible for generating the actual statistical information. In our approaches, we use a fairly simple filter function that computes the success rates concerning the two error classes described above (exception errors and errors in the binary representation). After filtering, the data is passed to an output module that translates the information into a format which

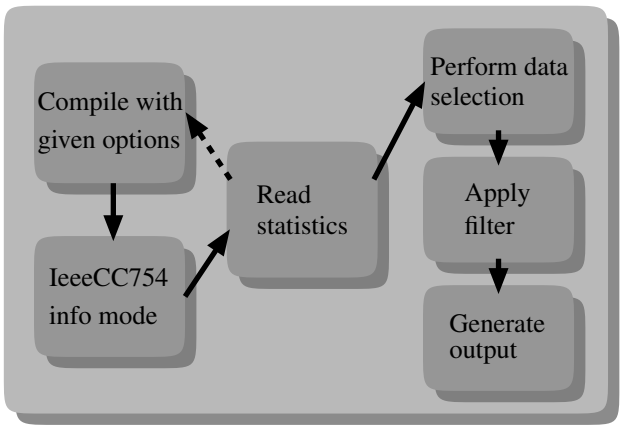


Figure 2. Generation framework

is recognized by the target grid information system (or e. g. into plain text format for testing purposes).

In the following, we present three approaches to publish helpful information in a grid context. Instead of describing the actual output that would be produced by an appropriate output module, we outline what type of information the variants will generate.

4.3. Variant 1: Hardware

The first variant generates information on the IEEE 754 compliance of the *hardware*. More specifically, it considers only the four basic arithmetic operations $+$, $-$, $*$ and $/$ and computes the percentage of successful tests with regard to the error classification (as described above). The tests may be performed by any compiler which maps the four operations to hardware functions with optimization turned off. Information will be displayed for each operation along with total results. As the actual calculations in the testing stage are performed in hardware, no information on the compiler used will be printed. Information on the hardware (processor type etc.) will not be generated as they should be present in the grid information system anyway.

The information generated by this approach might be used as another security measure to detect faulty processors in a grid. Another use case incorporates resource managers: It is possible to evaluate the numerical information and execute numerically demanding jobs only on platforms where the info mode information is present and a certain success rate required by the user is given (i. e. when a certain percentage of the tests was completed without errors).

4.4. Variant 2: Compiler options

While originally developed to publish information into a grid information system, the IeeeCC754 info mode has an interesting side use: For a given combination of platform and compiler, it is possible to evaluate different sets of optimization flags and choose the set that fits best (according to some criteria). So we can take our favorite compiler options, run IeeeCC754 info mode with these options and retrieve information on how the optimizing process affects floating-point reliability. To take things further, one might try to find a set of optimization flags that yields the highest possible performance while retaining the desired level of numerical stability (as far as floating-point calculations are concerned). This process is described in detail in Section 5, for our second info mode variant we take a slightly different approach.

IeeeCC754 standard mode testing results show that usually only a small percentage of errors is encountered. However, the requirement for small error rates is that optimization is disabled. As already mentioned, certain aspects of IEEE compliant floating-point arithmetic will be dropped at higher optimization levels due to the computational costs. An examination of the error rates in correlation to optimization levels reveals that typically only few errors are introduced until reaching a certain optimization level. From this point on, hundreds of errors will be encountered, sometimes to the extent that not a single (decimal) digit of a result is correct. Interestingly, these error rates behave quite differently when distinguishing once again between errors related to exceptions or to the binary representation: When considering exceptions, even low optimization levels usually generate high error rates, whereas error percentages with regard to the binary representation of results increase at a much slower rate. As an example, when using g++ 3.3 and only counting errors that relate to the binary representation, error rates range from 2% to 5% after compilation with `-O0`, `-O1` or `-O2` whereas using `-O3` results in an error rate of 40%. Regarding the total number of errors encountered (including exception flag errors), errors rates increase to 25% and 60% respectively.

Furthermore, compilers are able to produce code specifically tuned to the different types of processors and floating-point units (and, nowadays, multimedia extensions that feature fast floating-point implementations). This not only increases application performance, but also affects the quality of floating-point calculations, unfortunately not always in a positive sense.

These observations lead to our second approach: For a given compiler, we compute a suggestion which compiler options should be used under the assumptions that

- the highest possible performance should be obtained,
- the user does not rely on the implementation raising exception flags,
- the rounding mode will not be switched by the user,
- the error rate should be very small.

The information that will actually be displayed for a grid

node would consist of recommended options for some current and widely used compilers, e. g. g++ 3.3 and g++ 4.0. While it can not be guaranteed that the recommendation is the optimal set of options, users get at least a hint which set of compiler options yields a reasonably good result.

4.5. Variant 3: Fingerprint

While for numerically sensitive applications it is highly desirable to check every grid node via the checksum mode before executing the application, this approach is hardly feasible in a large scale grid due to the overhead that is generated when transferring the necessary data (the IeeeCC754 executable, testsets, and the local checksum). Therefore, our third variant with regard to grid information systems consists of a mixture of checksum and info modes. The idea is to store a "numerical fingerprint" of a platform on the grid so that resource management systems can use this information to dispatch jobs with special numerical requirements. From the considerations outlined so far, it is clear that there are some serious challenges related to this approach:

- Fingerprints can be stored only for a very limited number of compilers.
- It is not feasible to supply fingerprints for a wide range of compiler options, so one would have to define a small set of standard configurations of compiler options (e. g. only "`-O0`").
- This approach can only work if the release of IeeeCC754 and the testset used are identical; otherwise there would be no possibility to draw conclusions from comparing checksums. This obstacle can be overcome by defining fixed testsets and assigning version numbers both to IeeeCC754 and the testsets. These version numbers must be stored together with the fingerprint.
- The checksum must be small, i. e. a solution like the MD5- or SHA-hashes as proposed in Section 3 must be used. Here, the same considerations related to the computing of hash values apply – there is a small chance that identical hash values might be generated for different checksums.

Under these circumstances, it becomes possible to provide for a means of security similar to the checksum mode without transferring and executing IeeeCC754 for every submitted job. Thus no jobs must be canceled because of numerical incompatibility, they will not be submitted at all. The drawback is that the number of matching hosts might be small, and fingerprints must be supported by the resource broker in the first place.

Although using fingerprints proves valuable as a security measure (cf. Section 6), there remains need for research: To allow for a wider range of compatible grid nodes and more flexibility in general, it would be desirable to rate the quality of a node in a (ideally user-readable) way that allows for some "fuzziness" in the resource selection process. Unfortunately, all approaches considered so far do not yield a confidence level beyond that of variant 1. Especially, the main advantage of the checksum mode (a guarantee that a grid node behaves identical at least on a comprehensive testset) is lost. Future developments will concentrate on investigating the possibilities in this direction.

So far, we have realized proof-of-concept implementations of the described approaches that use MDS2 (the LDAP-based information system of the Globus Toolkit version 2 [13]). Later on it is planned to provide output modules for MDS4 (the information system of Globus Toolkit version 4 which is based on web-services) and Unicore [14].

5. Optimization of compiler options

The number and kind of errors found when executing any variant of IeeeCC754 is highly dependent on the optimization options passed to the compiler. The higher the optimization level, the more features of IEEE 754 arithmetic will be dropped as they usually degrade performance (as discussed before). Unfortunately, as long as there is no further knowledge available, the only way to guarantee the best possible floating-point reliability is to turn off any optimization whatsoever. On the other hand, using high optimization levels can result in drastically reduced computing times. We provide for a means to automatically optimize the set of compiler options with the following approach:

As explained in the last section, the framework built around the IeeeCC754 info mode enables the user to evaluate any number of compiler options settings. The optimization mode takes a (possibly large) choice of compiler options and executes the IeeeCC754 info mode for all possible combinations of options. Afterwards, every run is evaluated according to the specified filter settings. We implemented a supplementary *framework mode* that, in addition to calculating the error rate of a specific test run, benchmarks application performance by measuring the runtime of IeeeCC754. To get reliable results, a few runs are executed with identical settings and the average time is computed. All generated information is stored together with the setting of compiler options.

When all combinations have been processed, the choice of the best setting can be simply done by sorting. A weighting of the different values is achieved by supplying an appropriate comparison operator. We use the following order:

1. Success rate with regard to the binary result (larger is better).
2. Runtime (smaller is better).
3. Number of options (smaller is better).
4. Success rate regarding exception flags (larger is better).

The rationale behind this approach is the assumption that numerical reliability is considered the highest goal, closely followed by high performance. The reason for considering the number of compiler options is that when two sets of compiler options yield the same level of performance according to the first two criteria, it is simply more convenient to use the set with fewer options. If any two sets show the same performance even after comparing the number of options, finally the reliability values concerning exceptions flags are considered.

While the information gained by applying the described approach is valuable in itself, it should be noted that it can only yield general information with respect to performance. The main caveat is that the same set of compiler options can result in substantially different performances for different target applications. This simple observation somehow puts the results into perspective. As a remedy, instead of timing IeeeCC754, it is possible to record the runtimes of the actual application for which an "optimal" set of compiler options is sought for. In this mode the optimization framework takes a set of compiler options, compiles IeeeCC754 *and* the target application with these options, generates the statistical data related to numerical quality by running IeeeCC754 and measures the runtime of the application. In this manner it becomes possible to find a set of compiler options that yields very good performance of the application while guaranteeing the highest possible reliability with respect to floating-point operations. However, for this approach it is necessary that the application completes within a reasonable period of time – it is infeasible to test a broad range of possible combinations of compiler options when the application takes several hours (or even more than a few minutes) to complete. A possible remedy is to benchmark runtimes with a short running "toy" example application that resembles the target application more closely than IeeeCC754.

6. Test results

The tests were performed on two grids, the first being a small research and test grid at our department, the second the LCG (LHC Computing Grid) which provides the analysis infrastructure for CERN's Large Hadron Collider [9]. Our own test grid consists of several Linux workstations, all running some version of SuSE Linux, with hardware platforms ranging from Intel Pentium III and IV to AMD Athlon 32- and 64-bit processors. The LCG currently involves more than 200 sites in over 30 countries worldwide [6], the platforms deployed on the sites being too numerous to list.

Test results for the IeeeCC754 checksum mode are shown in table 1. The "local system" consisted of a Linux workstation with an Intel Pentium IV 2.8 GHz processor and 1 GB of memory (running SuSE Linux 8.2). All IeeeCC754 info mode runs were also performed on this system. In all following tests we only regard test vectors for single and double precision with rounding to nearest (since this is the configuration an application developer will encounter by default).

		-O0	-O1	-O2	-O3
g++ 3.3	local	1	1	1	100%
	LCG	2-3%	2-3%	2-3%	100%
g++ 4.0	local	0%	0%	100%	100%
	LCG	n/a	n/a	n/a	n/a

Table 1. Error rates (checksum mode)

For the checksum mode tests, the GNU compiler g++ version 3.3 was used. Four IeeeCC754 executables with different degrees of optimization were compiled (using `-O0`, `-O1`, `-O2`, and `-O3` resp.). The tests for g++ 3.3 (with low optimization levels) completed quite satisfactorily: Only 2-3% out of several hundred job runs on the LCG failed due to a different checksum. In our local test grid, only one job failed due to a quite outdated version of Linux on that system. These failures emphasize the need for measures of numerical qualities on a grid. As for the executable with the highest optimization level (the one compiled with `-O3`), not a single job could be successfully completed on either grid. Further investigation will be necessary before conclusions can be drawn, but the results show that the performance gained by using high levels of optimization comes at a price – by losing (some or all) IEEE 754 conformity when executed locally and by losing any predictability when submitting applications into a distributed environment. For g++ 4.0, the local tests yielded results similar to those with g++ 3.3 except that even for `-O2` all checksums were different. In the LCG, all submitted IeeeCC754 checksum executables compiled by g++ 4.0 failed to run. Analysis of the error logs revealed that the failure was due to missing libraries – starting from version 3.4 g++ uses a newer version of `libstdc++` which is currently not installed on the LCG.

Summarizing test results for the IeeeCC754 info mode runs is quite difficult as the different variants aim at providing information on a specific grid host. To get a feel for what information can be expected, we give examples generated on our "local system". For variant 1 (cf. Section 4.3) an example output in MDS2/LDAP format is shown in figure 3. Here, "success" stands for the percentage of correctly computed results compared to the total number of operations, while "result" gives the success rate of runs when only regarding the returned floating-point value. The high number of errors is due to the internal use of a higher precision format for intermediate calculations on Intel-compatible platforms.

```
Mds-IEEE-HW-all-success=97.28
Mds-IEEE-HW-all-result=98.99
Mds-IEEE-HW-add-success=99.49
Mds-IEEE-HW-add-result=99.49
Mds-IEEE-HW-mult-success=96.76
Mds-IEEE-HW-mult-result=99.12
Mds-IEEE-HW-div-success=95.37
Mds-IEEE-HW-div-result=97.96
```

Figure 3. Example MDS output

For variant 2 (cf. Section 4.4), we computed recommendations for an "optimal" set of compiler options (as would be shown in a grid information system). These recommendations were generated using a mixture of variant 2 and the optimization approach presented in Section 5: We specified some compiler options known to yield high performance and some that prevent the compiler from dropping certain aspects of IEEE 754 floating-point arithmetic. The resulting sets of options were "`-O2 -msse`" (with a success rate of 97.24% when disregarding errors due to exception flags and 76.34% otherwise) for g++ 3.3 and "`-O3 -mtune=pentium4 -ffloat-store`" for g++ 4.0 (with rates of 97.24 and 76.36%) respectively. Finally, showing test results (i. e. numerical fingerprints) for variant 3 of the IeeeCC754 info mode is omitted as the properties of the underlying numerical checksum have already been discussed in detail.

7. Summary

In this paper, we presented a couple of approaches to provide security measures with regard to floating-point arithmetic when executing applications in a grid context which demand a high level of numerical reliability. The checksum mode of IeeeCC754 provides a means to ensure that an application will only be executed on a grid node if this grid node supports the same floating-point behavior as the local system. The different variants of the info mode aim at storing information relevant to evaluating the numerical quality of a grid node in the grid information systems which can be exploited either automatically (e. g. by resource brokers) or manually (users or administrators trying to get measures of numerical reliability on a grid). Additionally, while not being directly grid-related, we showed an approach to tune a given application for optimum performance while retaining numerical quality. Together these approaches can help to ensure that numerical computations on a grid can be executed in a reliable manner.

The proposed IeeeCC754 checksum mode is about to be tested and integrated within the scope of the German D-GRID Initiative [3]. Furthermore, extensive tests of the info mode variants are planned.

Acknowledgements This work was partially supported by the BMBF (Bundesministerium für Bildung und Forschung) within the projects "Qualitätensicherung und Ressourcenoptimierung im GRID-Computing" and "Entwicklung von Anwendungen und Komponenten zur Datenauswertung in der Hochenergiephysik in einer nationalen e-Science Umgebung im Rahmen eines Verbundvorhabens" (D-GRID, HEP community, WP2).

References

[1] `http://www.ima.umn.edu/~arnold/455.f96/disasters.html`.

[2] A. Cuyt, P. Kuterna, B. Verdonk, and D. Verschaeren. Underflow revisited. *Calcolo*, 39:169–179, 2002.

[3] D-GRID Initiative. `http://www.d-grid.de/`.

[4] F. de Dinechin and G. Villard. High precision numerical accuracy for physics research. *Nuclear Instruments and Methods in Physics Research Section A*, 559(1):207–210, 2006.

[5] I. Foster and C. Kesselmann, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2003.

[6] GridCafé. `http://gridcafe.web.cern.ch/gridcafe/GridatCERN/LCG.html`.

[7] IEEE. ANSI/IEEE Std 754-1985, Standard for Binary Floating-Point Arithmetic, 1985.

[8] IEEE. ANSI/IEEE Std 854-1987, Standard for Radix-independent Floating-Point Arithmetic, 1987.

[9] LCG – The LHC Computing Grid Project. `http://lcg.web.cern.ch/LCG/`.

[10] NIST. Secure Hash Standard. Federal Information Processing Standards Publication 180-2, 2002.

[11] R. L. Rivest. The MD5 Message-Digest algorithm. `http://www.ietf.org/rfc/rfc1321.txt`, 1992.

[12] M. Taufer, D. Anderson, P. Cicotti, and C. L. Brooks. Homogeneous Redundancy: a Technique to Ensure Integrity of Molecular Simulation Results Using Public Computing. In *19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[13] The Globus Toolkit. `http://www.globus.org/toolkit`.

[14] Unicore. `http://unicore.sourceforge.net`.

[15] B. Verdonk, A. Cuyt, and D. Verschaeren. A precision- and range-independent tool for testing floating-point arithmetic I: basic operations, square root and remainder. *ACM Trans. Math. Softw.*, 27:92–118, 2001.

[16] B. Verdonk, A. Cuyt, and D. Verschaeren. A precision- and range-independent tool for testing floating-point arithmetic II: conversions. *ACM Trans. Math. Softw.*, 27:119–140, 2001.

An Interval Version of the Backward Differentiation (BDF) Method

Małgorzata Jankowska
Poznań University of Technology
Institute of Applied Mechanics
Piotrowo 3, 60-965 Poznań, Poland
Malgorzata.Jankowska@put.poznan.pl

Andrzej Marcinak
Poznań University of Technology
Institute of Computing Science
Piotrowo 2, 60-965 Poznań, Poland
&
Adam Mickiewicz University
Faculty of Mathematics and Computer Science
Umultowska 87, 61-614 Poznań, Poland
Andrzej.Marciniak@put.poznan.pl

Abstract

We present an interval version of the backward differentiation method for solving the initial value problem (IVP) for ordinary differential equations (ODEs). The method considered belongs to a group of the interval multistep methods. A number of the explicit and implicit interval multistep methods are tested on selected problems and the comparison is given.

1. Introduction

Studies on interval methods for solving the IVP started with the methods based on Taylor series (see e.g. Moore [20]). Since the problem seems very important for verified computing the considerations on such methods have been put forward in many papers and monographs (see e.g. Eijgeraam [2], Lohner [15], Rihm [22], [23], Berz, Makino and Hoefkens [1], [6], Gajda and Marcinaik [3], Kalmykov, Šokin and Juldašev [13], Marciniak [17], Marciniak and Szyszka [19], Nedialkov and Jackson [21], and Šokin [24]).

The research on the interval multistep methods began with the explicit interval methods of Adams-Bashforth type (EIAB) (see Jankowska and Marciniak [9], and Šokin [24]), two families of the implicit interval methods of Adams-Moulton type (IIAM) (see Jankowska and Marciniak [8], [11]) and two kinds of the interval predictor-corrector methods of Adams type (IIAPC1 and IIAPC2) (see Jankowska [7] and Jankowska and Marciniak [10]). Then, the explicit interval methods of Nyström type (EIN) and two families of implicit interval methods of Milne-Simpson type (IIMS) have been proposed (see Marciniak [18]).

The numerical analysis of conventional methods for solving the IVP shows the importance of the last group of multistep methods, i.e. the methods based on the backward differentiation formula (see Sec. 2). We propose the interval version of the explicit and implicit backward differentiation (BDF) methods (EIBDF and IIBDF) (see Sec. 3). The interval solutions obtained by such methods contain its errors (see Theorem 1 and Theorem 2 in Sec. 3). Furthermore, computer implementation of the interval BDF methods in floating-point interval arithmetic together with the representation of initial data in the form of machine intervals let us achieve the interval solutions which contains all possible numerical errors. The widths of the interval solutions obtained are also estimated (see Theorem 3 and Theorem 4 in Sec. 3). Finally, on the basis of some examples we compare a number of the multistep methods presented with the interval methods considered in our previous papers (see Sec. 4) and present some remarks (see Sec. 5).

2. The initial value problem and the conventional BDF method

The initial value problem (IVP) is of the form

$$y' = f(t, y), \quad y(0) = y_0, \tag{1}$$

where $t \in [0, \xi]$, $\xi \in \mathbb{R}$, $y = y(t) \in \mathbb{R}^N$, $f : [0, \xi] \times \mathbb{R}^N \to \mathbb{R}^N$. We will assume that the solution of (1) exists and is unique.

Now, let us choose a positive integer m and select the mesh points $t_0, t_1, \ldots, t_m$, where $t_n = nh$ for each $n = 0, 1, \ldots, m$ and $h = \xi/m$. Assume that the integer $k = 1, 2, \ldots$ will state how many approximations $y_{n-k}, y_{n-k+1}, \ldots, y_{n-1}$ of the exact solution at the previous k mesh points have to be known to get the approximation y_n $(n \geq k)$.

As is well known (see e.g. Hairer, Nørsett and Wanner [4] and Krupowicz [14]), the construction process of conventional backward differentiation method is based on the replacement of the exact solution $y(t)$ for $t \in [t_{n-k}, t_n]$ of the IVP by $P_k(t) + r(t)$, where $P_k(t)$ is the Newton backward-difference interpolation polynomial of degree k and $r(t)$ is the interpolation error term. Then, differentiating $y(t)$ we get

$$y'(t) = P_k'(t) + r'(t). \tag{2}$$

We assume that the polynomial $P_k(t)$ is such that $P_k(t_{n-i}) = y(t_{n-i})$, $i = 0, 1, \ldots, k$, and for $t = t_n + sh$, where h denotes the step size, $P_k(t)$ is of the form

$$\begin{aligned} P_k(t_n + sh) &= y(t_n) + s\nabla y(t_n) + \cdots + \\ &\quad + \frac{s(s+1)\cdots(s+k-1)}{k!}\nabla^k y(t_n), \end{aligned}$$

and the interpolation error $r(t)$ is given by the formula

$$r(t_n + sh) = y^{(k+1)}(\zeta(s))\, h^{k+1} \frac{s(s+1)\cdots(s+k)}{(k+1)!},$$

where $\zeta(s) \in [t_{n-k}, t_n]$.
We consider (2) for $t = t_n, t_{n-1}, \ldots, t_{n-k}$. Hence, we have

$$y'(t_{n-i}) = P_k'(t_{n-i}) + r'(t_{n-i}), \quad i = 0, 1, \ldots, k, \tag{3}$$

and performing the differentiation required, finally for $i = 0, 1, \ldots, k$ we get

$$\begin{aligned} y(t_n) &= \sum_{j=1}^{k} \alpha_{kj}^{(i)} y(t_{n-j}) \\ &\quad + h\beta_{ki}^{(i)} f(t_{n-i}, y(t_{n-i})) \\ &\quad + h^{k+1}\gamma_{ki}^{(i)} \psi(\eta_k, y(\eta_k)), \end{aligned} \tag{4}$$

where $\eta_k \in [t_{n-k}, t_n]$, $\psi(\eta_k, y(\eta_k)) \equiv f^{(k)}(\eta_k, y(\eta_k)) \equiv y^{(k+1)}(\eta_k)$, and

$$\alpha_{kj}^{(i)} = -\frac{(-1)^j \sum_{m=j}^{k} \binom{m}{j} \delta_{i,m}}{\sum_{m=0}^{k} \delta_{i,m}}, \tag{5}$$

$$\beta_{ki}^{(i)} = \frac{1}{\sum_{m=0}^{k} \delta_{i,m}}, \quad \gamma_{ki}^{(i)} = -\frac{\delta_{i,k+1}}{\sum_{m=0}^{k} \delta_{i,m}}, \tag{6}$$

$$\delta_{i,m} = \begin{cases} \frac{d}{ds}\frac{s(s+1)\cdots(s+m-1)}{m!}\big|_{s=-i}, & m \geq 1 \\ 0, & m = 0 \end{cases}. \tag{7}$$

In the numerical analysis of the conventional methods for solving the IVP it used to consider only two of $k+1$ different cases, i.e. when $i = 0$ or $i = 1$. Hence, for $i = 1$ we have

$$\begin{aligned} y(t_n) &= \sum_{j=1}^{k} \alpha_{kj} y(t_{n-j}) \\ &\quad + h\beta_{k1} f(t_{n-1}, y(t_{n-1})) \\ &\quad + h^{k+1}\gamma_{k1} \psi(\eta_k, y(\eta_k)), \end{aligned} \tag{8}$$

where

$$\alpha_{kj} = -\frac{(-1)^j \sum_{m=j}^{k} \binom{m}{j} \delta_{1,m}}{\sum_{m=0}^{k} \delta_{1,m}}, \tag{9}$$

$$\beta_{k1} = \frac{1}{\sum_{m=0}^{k} \delta_{1,m}}, \quad \gamma_{k1} = -\frac{\delta_{1,k+1}}{\sum_{m=0}^{k} \delta_{1,m}}, \tag{10}$$

and for $i = 0$ we have

$$\begin{aligned} y(t_n) &= \sum_{j=1}^{k} \overline{\alpha}_{kj} y(t_{n-j}) \\ &\quad + h\overline{\beta}_{k0} f(t_n, y(t_n)) \\ &\quad + h^{k+1}\overline{\gamma}_{k0} \psi(\eta_k, y(\eta_k)), \end{aligned} \tag{11}$$

where

$$\overline{\alpha}_{kj} = -\frac{(-1)^j \sum_{m=j}^{k} \binom{m}{j} \delta_{0,m}}{\sum_{m=0}^{k} \delta_{0,m}}, \tag{12}$$

$$\overline{\beta}_{k0} = \frac{1}{\sum_{m=0}^{k} \delta_{0,m}}, \quad \overline{\gamma}_{k0} = -\frac{\delta_{0,k+1}}{\sum_{m=0}^{k} \delta_{0,m}}. \tag{13}$$

The unknown values $y(t_{n-j})$, $j = 1, 2, \ldots, k$ in (8) and (11), $f(t_{n-1}, y(t_{n-1}))$ in (8) and $f(t_n, y(t_n))$ in (11) are replaced with the approximations y_{n-j}, $j = 1, 2, \ldots, k$, $f_{n-1} = f(t_{n-1}, y_{n-1})$ and $f_n = f(t_n, y_n)$, respectively. Furthermore, we neglect the error terms $h^{k+1}\gamma_{k1}\psi(\eta_k, y(\eta_k))$ and $h^{k+1}\overline{\gamma}_{k0}\psi(\eta_k, y(\eta_k))$. This leads to the following formulas that determine the algorithm of the explicit conventional k-step backward differentiation method:

$$y_n = \sum_{j=1}^{k} \alpha_{kj} y_{n-j} + h\beta_{k1} f(t_{n-1}, y_{n-1}), \tag{14}$$

and the implicit one given below

$$y_n = \sum_{j=1}^{k} \overline{\alpha}_{kj} y_{n-j} + h\overline{\beta}_{k0} f(t_n, y_n). \tag{15}$$

Let us note that the explicit conventional one step BDE method is in fact the conventional one-step Adams-Bashforth method. The explicit conventional two step BDF method is the conventional two step Nyström method (the similar conclusion does not hold for the explicit interval two step method of BDE type). Furthermore, the explicit conventional BDF methods for $k \geq 3$ are unfortunately unstable.

3. An interval version of the BDF method

Let us denote Δ_t and Δ_y as sets in which the function $f(t,y)$ of the IVP (1) is defined as follows:

$$\Delta_t = \{t \in \mathbb{R} : 0 \leq t \leq \xi,\ \xi \in \mathbb{R}\},$$

$$\Delta_y = \Big\{y = (y_1, y_2, \ldots, y_N)^T \in \mathbb{R}^N : \underline{b}_i \leq y_i \leq \overline{b}_i,\ \underline{b}_i, \overline{b}_i \in \mathbb{R},\ i = 1,2,\ldots,N\Big\}.$$

Let $F(T,Y)$ and $\Psi(T,Y)$ be interval extensions of $f(t,y)$ and $\psi(t, y(t)) \equiv f^{(k)}(t, y(t)) \equiv y^{(k+1)}(t)$, respectively.

We also assume that

- $F(T,Y)$ is defined and continuous for all $T \subset \Delta_t$ and $Y \subset \Delta_y$ (for a concept of continuity of interval function see e.g. Moore [20]),
- $F(T,Y)$ is monotonic with respect to inclusion, i.e.

$$T_1 \subset T_2 \wedge Y_1 \subset Y_2 \Rightarrow F(T_1, Y_1) \subset F(T_2, Y_2),$$

- for each $T \subset \Delta_t$ and for each $Y \subset \Delta_y$ there exists a constant $L > 0$ such that

$$d(F(T,Y)) \leq L(d(T) + d(Y)),$$

where $d(A)$ denotes the width of A
(if $A = (A_1, A_2, \ldots, A_N)^T$ then $d(A)$ is defined by $d(A) = \max\limits_{i=1,2,\ldots,N} d(A_i)$),

- $\Psi(T,Y)$ is defined for all $T \subset \Delta_t$ and $Y \subset \Delta_y$,
- $\Psi(T,Y)$ is monotonic with respect to inclusion.

The interval version of explicit BDF method we define as follows

$$\begin{aligned} Y_n = \sum_{j=1}^{k} \alpha_{kj} Y_{n-j} &+ h\beta_{k1} F(T_{n-1}, Y_{n-1}) \\ &+ h^{k+1}\gamma_{k1}\Psi(T_{n-k} + [0, kh], \\ &\qquad Y_{n-k} + [0, kh]\, F(\Delta_t, \Delta_y)), \end{aligned} \tag{16}$$

for $n = k, k+1, \ldots, m$,

where $h = \xi/m$, $t_i = ih \in T_i$, $i = 0,1,\ldots,m$, α_{kj}, $j = 1,2,\ldots,k$, are given by (9), β_{k1} and γ_{k1} are given by (10).

In particular, for a given k from (16) we have the following methods:

- $k = 1$ (the same formula as for the explicit interval one step method of Adams-Bashforth type [9])

$$\begin{aligned} Y_n &= Y_{n-1} + hF(T_{n-1}, Y_{n-1}) \\ &\quad + \frac{1}{2}h^2\Psi(T_{n-1} + [0,h], \\ &\qquad Y_{n-1} + [0,h]\, F(\Delta_t, \Delta_y)), \end{aligned}$$

- $k = 2$

$$\begin{aligned} Y_n &= Y_{n-2} + 2hF(T_{n-1}, Y_{n-1}) \\ &\quad + \frac{1}{3}h^3\Psi(T_{n-2} + [0,2h], \\ &\qquad Y_{n-2} + [0,2h]\, F(\Delta_t, \Delta_y)), \end{aligned}$$

- $k = 3$

$$\begin{aligned} Y_n &= \frac{1}{2}(-3Y_{n-1} + 6Y_{n-2} - Y_{n-3}) \\ &\quad + 3hF(T_{n-1}, Y_{n-1}) \\ &\quad + \frac{1}{4}h^4\Psi(T_{n-3} + [0,3h] \\ &\qquad Y_{n-3} + [0,3h]\, F(\Delta_t, \Delta_y)). \end{aligned}$$

The interval version of implicit BDF method can be given by the formula

$$\begin{aligned} Y_n = \sum_{j=1}^{k} \overline{\alpha}_{kj} Y_{n-j} &+ h\overline{\beta}_{k0} F(T_n, Y_n) \\ &+ h^{k+1}\overline{\gamma}_{k0}\Psi(T_{n-k} + [0, kh], \\ &\qquad Y_{n-k} + [0, kh]\, F(\Delta_t, \Delta_y)), \end{aligned} \tag{17}$$

for $n = k, k+1, \ldots, m$,

where $h = \xi/m$, $t_i = ih \in T_i$, $i = 0,1,\ldots,m$, $\overline{\alpha}_{kj}$, $j = 1,2,\ldots,k$, are given by (12), $\overline{\beta}_{k0}$ and $\overline{\gamma}_{k0}$ are given by (13).

In particular, for a given k from (17) we have the following methods:

- $k=1$

$$\begin{aligned} Y_n &= Y_{n-1}+hF\left(T_n,Y_n\right) \\ &\quad -\frac{1}{2}h^2\Psi\left(T_{n-1}+[0,h]\right. \\ &\qquad \left. Y_{n-1}+[0,h]\,F\left(\Delta_t,\Delta_y\right)\right), \end{aligned}$$

- $k=2$

$$\begin{aligned} Y_n &= \frac{1}{3}\left(4Y_{n-1}-Y_{n-2}\right)+\frac{2}{3}hF\left(T_n,Y_n\right) \\ &\quad -\frac{2}{9}h^3\Psi\left(T_{n-2}+[0,2h]\right. \\ &\qquad \left. Y_{n-2}+[0,2h]\,F\left(\Delta_t,\Delta_y\right)\right), \end{aligned}$$

- $k=3$

$$\begin{aligned} Y_n &= \frac{1}{11}\left(18Y_{n-1}-9Y_{n-2}+2Y_{n-3}\right) \\ &\quad +\frac{6}{11}hF\left(T_n,Y_n\right) \\ &\quad -\frac{3}{22}h^4\Psi\left(T_{n-3}+[0,3h]\right. \\ &\qquad \left. Y_{n-3}+[0,3h]\,F\left(\Delta_t,\Delta_y\right)\right). \end{aligned}$$

In each step of the interval version of the implicit BDF method we have to solve a system of nonlinear interval equations of the following form

$$Y=G\left(T,Y\right),$$

where

$$\begin{gathered} T\in\mathbb{I}\left(\Delta_t\right)\subset\mathbb{IR}, \\ Y=\left(Y_1,Y_2,\ldots,Y_n\right)^T\in\mathbb{I}\left(\Delta_y\right)\subset\mathbb{IR}^N, \\ G:\mathbb{I}\left(\Delta_t\right)\times\mathbb{I}\left(\Delta_y\right)\rightarrow\mathbb{IR}^N, \end{gathered}$$

and $\mathbb{IR}$ and $\mathbb{IR}^N$ represent the sets of all real intervals and all n-dimensional real interval vectors, respectively. We use the notation $\mathbb{I}\left(A\right)$, for an interval $A=[\underline{x},\overline{x}]$, to mean the set of intervals which are contained in A, i.e. $\mathbb{I}\left(A\right)=\{[x,y]:\underline{x}\leq x\leq y\leq\overline{x}\}$. In other words, $\mathbb{I}\left(A\right)$ is the set of subintervals of A. If we assume that the interval function G is a contraction mapping, then the well known fixed-point theorem implies that the iteration process

$$Y^{(l+1)}=G\left(T,Y^{(l)}\right),\quad l=0,1,\ldots, \tag{18}$$

converges to an unique element Y^*, i.e. $\lim\limits_{l\rightarrow\infty}Y^{(l)}=Y^*$, for an arbitrary choice of $Y^{(0)}\in\mathbb{I}\left(\Delta_y\right)$.

The iteration process (18) for the interval version of the implicit BDF method (17) is of the form

$$\begin{aligned} Y_n^{(l+1)} &= \sum_{j=1}^{k}\overline{\alpha}_{kj}Y_{n-j}+h\overline{\beta}_{k0}F\left(T_n,Y_n^{(l)}\right) \\ &\quad +h^{k+1}\overline{\gamma}_{k0}\Psi\left(T_{n-k}+[0,kh]\right. \\ &\qquad \left. Y_{n-k}+[0,kh]\,F\left(\Delta_t,\Delta_y\right)\right), \\ l &= 0,1,\ldots, \end{aligned} \tag{19}$$

and we usually choose $Y_n^{(0)}=Y_{n-1}$, where Y_{n-1} is the solution obtained at t_{n-1}. Let us note that the intervals Y_{n-j} are constant during the iteration process. In practice we stop the iteration process after some accuracy given beforehand.

For the interval version of explicit and implicit methods of BDF type we can prove that the exact solution of the IVP belongs to the intervals obtained with the methods considered (see Theorem 1 and Theorem 2). Furthermore, we can estimate the widths of the interval solutions (see Theorem 3 and Theorem 4).

Theorem 1 *If $y\left(0\right)\in Y_0$, $y\left(t_i\right)\in Y_i$ for $i=1,2,\ldots,k-1$, then for the exact solution $y\left(t\right)$ of the IVP* (1) *we have*

$$y\left(t_n\right)\in Y_n,$$

for $n=k,k+1,\ldots,m$, where $Y_n=Y\left(t_n\right)$ are obtained from the method (16)*.*

Theorem 2 *If $y\left(0\right)\in Y_0$, $y\left(t_i\right)\in Y_i$ for $i=1,2,\ldots,k-1$, then for the exact solution $y\left(t\right)$ of the IVP* (1) *we have*

$$y\left(t_n\right)\in Y_n,$$

for $n=k,k+1,\ldots,m$, where $Y_n=Y\left(t_n\right)$ are obtained from the method (17)*.*

Theorem 3 *If the intervals Y_n for $n=0,1,\ldots,k-1$ are known, $t_i=ih\in T_i$ for $i=0,1,\ldots,m$, $h=\xi/m$, and Y_n for $n=k,k+1,\ldots,m$ are obtained from the method* (16)*, then*

$$\begin{aligned} d\left(Y_n\right) &\leq A\max_{q\,=\,0,1,\ldots,k-1}d\left(Y_q\right) \\ &\quad +B\max_{j\,=\,k-1,k,\ldots,m-1}d\left(T_j\right)+Ch^k, \end{aligned}$$

where the constants A, B and C are independent of h.

Theorem 4 *If the intervals Y_n for $n=0,1,\ldots,k-1$ are known, $t_i=ih\in T_i$ for $i=0,1,\ldots,m$,*

$$h=\xi/m,\quad 0<h\leq h_0,$$

where

$$h_0<\frac{1}{L\overline{\beta}_{k0}},\quad L>0,$$

and Y_n *for* $n = k, k+1, \ldots, m$ *are obtained from the method* (17)*, then*

$$d(Y_n) \leq A \max_{q=0,1,\ldots,k-1} d(Y_q) + B \max_{j=k,k+1,\ldots,m} d(T_j) + Ch^{k+1},$$

where the constants A*,* B *and* C *are independent of* h*.*

For the proofs of the above theorems see [12].

4. Numerical examples

In this section we present some numerical results for the interval multistep methods mentioned in this paper. We tested the algorithms using our *IntervalArithmetic* unit written in the Delphi Pascal language and in the C++ language as well. Applying apporiate rounding modes the units make it possible to represent the input data in the form of machine intervals, use the floating-point interval arithmetic and obtain the results in the form of proper intervals.

Example 1

Consider the test problem of the form

$$y' = 0.5y, \quad y(0) = 1. \tag{20}$$

The exact solution of (4.1) is known

$$y = e^{0.5t}. \tag{21}$$

We integrate (20) for $t \in [0,\ 1]$. Hence, $\Delta_t = [0,\ 1]$ and we take $Y(0) = [1,\ 1]$ and $\Delta_y = [0.98,\ 1.66]$.

Figure 1 shows a comparison of the widths of the interval solution $Y(1)$ obtained from the integration of the problem considered by the interval explicit methods, i.e. the interval version of explicit BDF method for $k = 1, 2$ and the explicit method of Nyström type for $k = 1, 2, \ldots, 6$ versus different values of the stepsize h.

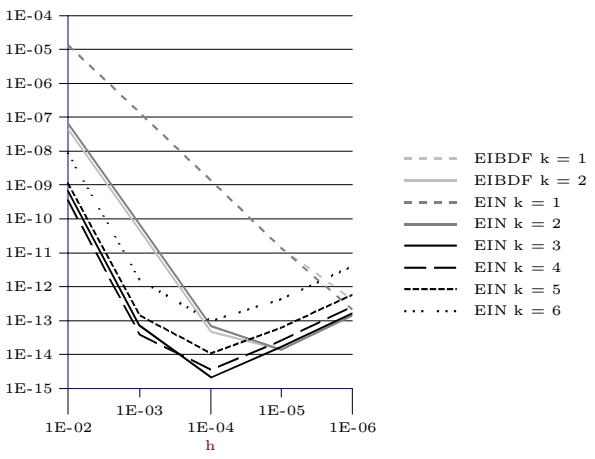


Figure 1. Widths of the interval solution $Y(1)$ obtained from the integration of (20) vs. the stepsize h.

Figure 2 gives a comparison of the explicit interval methods, i.e. the interval methods of Adams-Bashforth type, the interval methods of Nyström type and the interval version of explicit BDF method, and the implicit interval methods, i.e. the interval methods of Adams-Moulton type, the interval methods of Milne-Simpson type and the interval version of implicit BDF method for some values of parameter k and the stepsize $h =$ 5E-5.

Let us note that the solution obtained by the EIBDF methods for $k = 1$ and the EIN methods for $k = 1$, where $h =$ 1E-2, $h =$ 1E-3 and $h =$ 1E-4 have similar widths and are not distinguishable on Figure 1.

As one can see the interval version of explicit BDF method gives the interval solutions of acceptable widths only for the parameter $k = 1, 2$. For $k \geq 3$ the widths grow in an uncontrolled way and the reason of such behaviour can be the fact that the explicit conventional BDF method is unstable for $k \geq 3$ (see e.g. Hairer, Norsett and Wanner [4] and Krupowicz [14]). The interval version of implicit BDF method gives the acceptable interval solutions only for $k = 1$. If $k \geq 2$ we have the similar undesirable situation as for the explicit ones.

On the other hand let us note that for any value of the stepsize h the interval version of explicit BDF method gives the best results in the group of all explicit interval methods used for $k = 2$. For $k \geq 3$ we recommend the explicit interval methods of Nyström type rather then the explicit interval methods of Adams-Bashforth type.

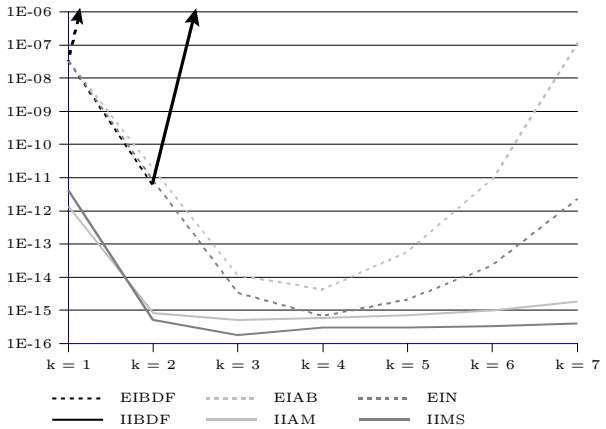


Figure 2. Widths of the interval solution $Y(1)$ obtained from the integration of (20) vs. the number k of method steps, where $h =$ 5E-4.

Let us also note that the implicit interval methods of Milne-Simpson type are the best methods in the class of the implicit interval multistep methods. We also recommend the interval predictor-corrector methods of Adams type (see Example 2). Unfortunately, the interval version of BDF

method did not meet our expectations but we found it essential to present the considerations mentioned in our paper.

Example 2

As an example of multidimensional IVP we consider the equations of motion of the Moon given by Hill (see e.g. Hill [5], Marciniak [16]). We assume that the origin of the frame is placed in the center of the Earth, the plane xy is in line with the Sun's orbit plane and the frame rotates with the constant angular velocity ν', where ν' denotes the mean motion of the Sun. Moreover, we assume that in the considered frame of reference the axis x goes across the center of the Sun and the Sun revolves around the Earth along the circular orbit. Finally, we take into account a small inclination of the Moon's orbit to the ecliptic.

Under such assumptions the equations of motion of the Moon (called the Hill equations) are of the form

$$\begin{aligned} \frac{d^2x}{d\tau^2} &= 2M\frac{dy}{d\tau} - \left(\frac{\kappa}{r^3} - 3M^2\right)x, \\ \frac{d^2y}{d\tau^2} &= -2M\frac{dx}{d\tau} - \frac{\kappa}{r^3}y, \end{aligned} \tag{22}$$

where $x = x(\tau)$, $y = y(\tau)$, $\tau = (\nu - \nu')(t - t_0)$, and $r = \sqrt{x^2 + y^2}$. Furthermore, the parameters M and κ in (22) are introduced as follows

$$M = \frac{\nu'}{\nu - \nu'}, \quad \kappa = G\,\frac{m_0 + m_1}{(\nu - \nu')^2}, \tag{23}$$

where ν is the mean motion of the Moon, G is the gravitational constant, and m_0, m_1 denote the masses of the Earth and the Moon, respectively.

The motion of the Moon (22) is given by the system of differential equations of the first order

$$\begin{aligned} \frac{du_1}{d\tau} &= u_3, \quad \frac{du_2}{d\tau} = u_4, \\ \frac{du_3}{d\tau} &= 2Mu_4 - \left(\frac{\kappa}{r^3} - 3M^2\right)u_1, \\ \frac{du_4}{d\tau} &= -2Mu_3 - \frac{\kappa}{r^3}u_2, \end{aligned} \tag{24}$$

with the initial conditions

$$\begin{aligned} u_1(0) = x_0, \;\; u_2(0) = y_0, \\ u_3(0) = v_{x0}, \;\; u_4(0) = v_{y0}, \end{aligned} \tag{25}$$

where

$$u_1 = x,\; u_2 = y,\; u_3 = \frac{dx}{d\tau},\; u_4 = \frac{dy}{d\tau},\; r = \sqrt{u_1^2 + u_2^2}.$$

Let us note that if $M = 0$ (i.e. the considerable part of perturbations is neglected), then the periodical solution, called a variational curve, is given by the formulas

$$\begin{aligned} u_1 = \kappa^{\frac{1}{3}}\cos(\tau), \;\; u_2 = \kappa^{\frac{1}{3}}\sin(\tau), \\ u_3 = -\kappa^{\frac{1}{3}}\sin(\tau), \;\; u_4 = \kappa^{\frac{1}{3}}\cos(\tau). \end{aligned} \tag{26}$$

At first we take the initial conditions (25) as follows

$$u_1(0) = 1, \;\; u_2(0) = 0, \;\; u_3(0) = 0, \;\; u_4(0) = 1, \tag{27}$$

and specify the parameters M and κ in (23) as $M = 0$, $\kappa = 1$. We integrate (24) with (27) for $\tau \in [0,\ 8]$. Hence, $\Delta_t = [0,\ 8]$ and we take $U_1(0) = [1,\ 1]$, $U_2(0) = [0,\ 0]$, $U_3(0) = [0,\ 0]$ and $U_4(0) = [1,\ 1]$. The IVP (24) with (27) has been solved in several stages. Hence, for each integration interval the appropriate set Δ_y has been specified.

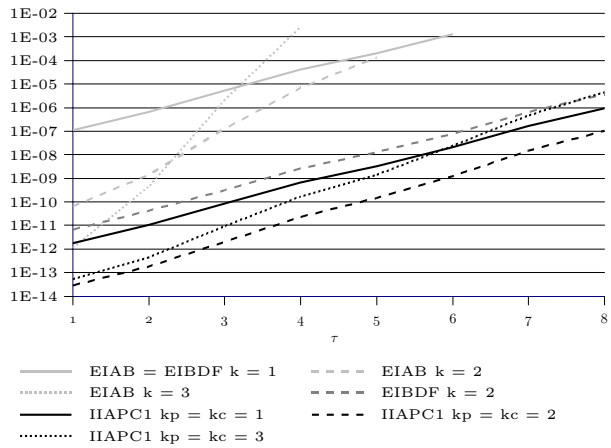


Figure 3. Widths of the interval solution $U_1(\tau)$ obtained from the integration of the Hill equations (24) with (27), with $M = 0$, $\kappa = 1$, by the EIAB methods for $k = 1, 2, 3$, the EIBDF methods for $k = 1, 2$ and the IIAPC1 methods for $k_p = k_c = 1, 2, 3$ ($k_p = k_c$ - the numbers of steps for predictor and corrector, respectively), where $h =$ 1E-5.

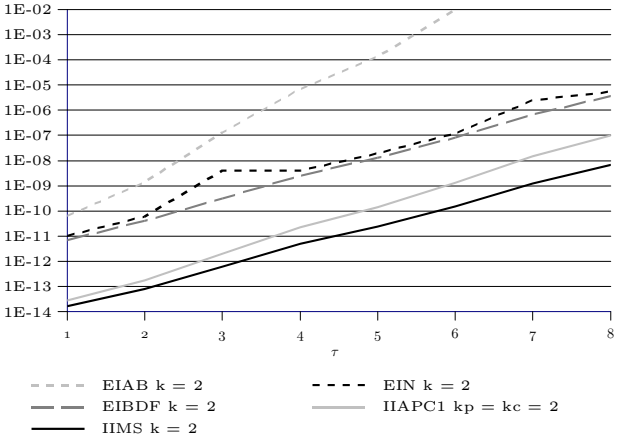


Figure 4. Widths of the interval solution $U_1(\tau)$ obtained from the integration of the Hill equations (24) with (27), with $M = 0$, $\kappa = 1$, by the EIAB method, the EIN method, the EIBDF method, the IIAPC1 method and the IIMS method, where $k = k_p = k_c = 2$ and $h =$ 1E-5.

In the Figure 3 the widths of the interval solutions $U_1(\tau)$ obtained by the interval explicit methods of Adams-Bashforth type and the interval predictor-corrector methods of Adams type together with the results given by the interval version of explicit BDF method for $k = 1, 2$ where $h =$ 1E-5, are presented. The Figure 4 shows a comparison of the widths of the interval solution $U_1(\tau)$ obtained by the interval explicit and implicit multistep methods considered for $k = 2$ and $h =$ 1E-5.

Now, let us take (see Hill [5], Marciniak [16]) the initial conditions (25) as follows

$$\begin{aligned} u_1(0) &= -0.9447782,\ u_2(0) = -0.2673999, \\ u_3(0) &= 0.3286969,\ u_4(0) = -0.9500594, \end{aligned} \quad (28)$$

and specify the parameters M and κ in (23) as
$M = 0.080848933808312,\ \kappa = 1.171418459184516$.

We integrate (24) with (28) for $\tau \in [0,\ 8]$. We have $\Delta_t = [0,\ 8]$ and

$$\begin{aligned} U_1 &= [\bigtriangledown(-0.9447782),\ \bigtriangleup(-0.9447782)], \\ U_2 &= [\bigtriangledown(-0.2673999),\ \bigtriangleup(-0.2673999)], \\ U_3 &= [\bigtriangledown(0.3286969),\ \bigtriangleup(0.3286969)], \\ U_4 &= [\bigtriangledown(-0.9500594),\ \bigtriangleup(-0.9500594)], \end{aligned}$$

where $\bigtriangledown(\cdot)$ means the number obtained by downrounding and $\bigtriangleup(\cdot)$ - by uprounding. Furthermore, the set Δ_y is specified as previous for each integration interval and the interval representation of M and κ are used.

The results obtained from the integration of the Hill equations (24) with (28) only by the explicit interval methods for the parameter $k = 2$ are given in Figure 5. Furthermore, in Figure 6 the positions u_1, u_2, and the velocities u_3, u_4 of the Moon as functions of τ, where $\tau \in [0,\ 8]$ are given. The orbit is shown in Figure 7.

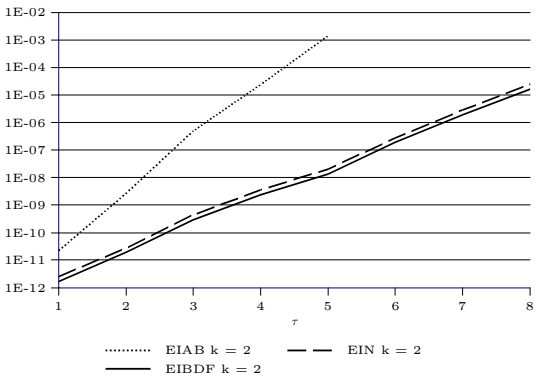


Figure 5. Widths of the interval solution $U_1(\tau)$ obtained from the integration of the Hill equations (24) with (28), with $M \approx 0.08$, $\kappa \approx 1.17$, by the EIAB method, the EIN method, and EIBDF method, where $k = 2$ and $h =$ 1E-5.

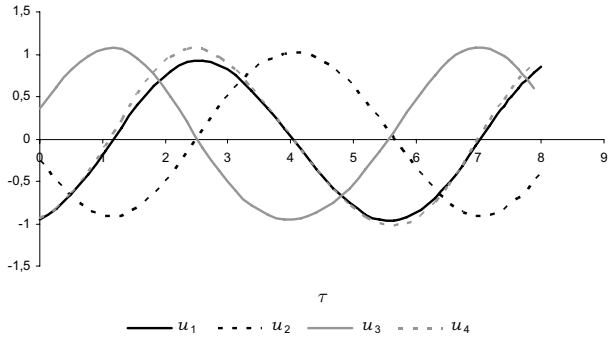


Figure 6. Positions u_1, u_2, and velocities u_3, u_4 of the Moon, obtained from the integration of the Hill equations (24) with (28) for $\tau \in [0,\ 8]$, with $M \approx 0.08$, $\kappa \approx 1.17$, by the IIAPC1 method for $k_p = k_c = 2$, where $h =$ 1E-5.

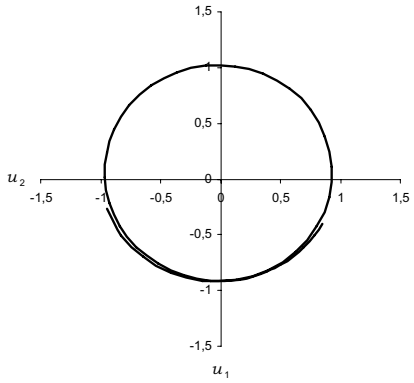


Figure 7. The orbit obtained from the integration of the Hill equations (24) with (28) for $\tau \in [0,\ 8]$, with $M \approx 0.08$, $\kappa \approx 1.17$, by the IIAPC1 method for $k_p = k_c = 2$, where $h =$ 1E-5.

5. Conclusions

In the paper the interval version of BDF method is proposed. Furthermore, all the interval multistep methods presented by authors are compared and the efficiency of the appropriate algorithms is given. From the analysis of the numerical results we can draw some conclusions.

First, as one might suppose, the implicit interval methods, except of the interval version of the BDF method, yield better results, i.e. interval solutions of a smaller width, than the explicit ones applied with the same number k of method steps and the same stepsize h.

Second, the interval methods of BDF type are not so good as other multistep methods. For the same number of steps the explicit interval methods of Nyström type are

somewhat better than the methods of Adams-Bashforth type (and also then the explicit interval methods of BDF type for the parameter $k \geq 3$). The interval version of explicit BDF method is the best in the class of all explicit interval methods for $k = 1, 2$. Similarly for the same number of steps the implicit interval methods of Milne-Simpson type give somewhat better result than the interval methods of Adams-Moulton type (and in many cases the implicit interval methods of BDF type).

Finally, for the explicit and implicit interval multistep methods considered we see that if we take the greater number k method steps, then we obtain the interval solutions with the smaller widths.

The increase in the number k of method steps for the same stepsize h contributes to the decrease in the widths of the interval solutions. A similar effect can be observed if we reduce the stepsize h for the same value of parameter k. This behaviour is true mainly for short integration intervals. Otherwise for each particular IVP the suitable interval method with the appropriate number k of method steps and the stepsize h should be chosen to get the best acceptable result. Moreover, for a given stepsize there exists an optimal number of method steps, and for a given number of method steps the optimal stepsize can be found.

References

[1] M. Berz and K. Makino. Verified integration of ODEs and flows with differential algebraic methods on taylor models. *Reliable Computing*, 4(4):361–369, 1998.

[2] P. Eijgenraam. The Solution of Initial Value Problems Using Interval Arithmetic. *Mathematical Centre Tracks*, 144, 1981.

[3] K. Gajda, A. Marciniak, and B. Szyszka. Three- and Four-Stage Implicit Interval Methods of Runge-Kutta Type. *Computational Methods in Science and Technology*, 6:41–59, 2000.

[4] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I - Nonstiff Problems*. Springer-Verlag, Berlin, Heidelberg, 1987.

[5] G. W. Hill. Researches in the lunar theory. *Am. J. Math. I*, 1878.

[6] J. Hoefkens, M. Berz, and K. Makino. Controlling the Wrapping Effect in the Solution of ODEs for Asteroids. *Reliable Computing*, 8:21–41, 2003.

[7] M. Jankowska. *Interval Multistep Methods of Adams Type and their Implementation in the C++ Language*. PhD thesis, Poznań University of Technology, 2006.

[8] M. Jankowska and A. Marciniak. Implicit Interval Multistep Methods for Solving the Initial Value Problem. *Computational Methods in Science and Technology*, 8(1):17–30, 2002.

[9] M. Jankowska and A. Marciniak. On Explicit Interval Methods of Adams-Bashforth Type. *Computational Methods in Science and Technology*, 8(2):46–57, 2002.

[10] M. Jankowska and A. Marciniak. Preliminaries of the IMM System for Solving the Initial Value Problem by Interval Multistep Methods [in Polish]. *Pro Dialog*, 10:117–134, 2005.

[11] M. Jankowska and A. Marciniak. On Two Families of Implicit Interval Methods of Adams-Moulton Type. *Computational Methods in Science and Technology*, 12(2):109–113, 2006.

[12] M. Jankowska and A. Marciniak. On the Interval Methods of the BDF Type for Solving the Initial Value Problem. *Pro Dialog*, 22:39–59, 2007.

[13] S. A. Kalmykov, J. I. Šokin, and E. C. Juldašev. Solving ordinary differential equations by interval methods [in Russian]. *Doklady AN SSSR*, 230(6), 1976.

[14] A. Krupowicz. *Numerical Methods of Initial Value Problems of Ordinary Differential Equations [in Polish]*. PWN, Warsaw, 1986.

[15] R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. W. Kaucher, U. W. Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*. Wiley-Teubner Series in Computer Science, Stuttgart, 1987.

[16] A. Marciniak. *Numerical Solutions of the N-body Problem*. Reidel, Dordrecht, 1985.

[17] A. Marciniak. Implicit Interval Methods for Solving the Initial Value Problem. *Numerical Algorithms*, 37:241–251, 2004.

[18] A. Marciniak. On Multistep Interval Methods for Solving the Initial Value Problem. *Journal of Computational and Applied Mathematics*, 199:229–237, 2007.

[19] A. Marciniak and B. Szyszka. One- and Two-Stage Implicit Interval Methods of Runge-Kutta Type. *Computational Methods in Science and Technology*, 5:53–65, 1999.

[20] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[21] N. S. Nedialkov and K. R. Jackson. An interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. *Reliable Computing*, 5(3):289–310, 1999.

[22] R. Rihm. Interval methods for initial value problems in ODE's. In J. Herzberger, editor, *Topics in Validated Computations, Proceedings of the IMACS-GAMM International Workshop on Validated Computations, Oldenburg, Germany*, 1994.

[23] R. Rihm. On a class of enclosure methods for initial value problems. *Computing*, 53:369–377, 1994.

[24] J. I. Šokin. *Interval Analysis [in Russian]*. Nauka, Novosibirsk, 1981.

A workload analysis tool for discrete-time semi-Markovian servers

Sebastian Kempken
Abteilung Informatik
Universität Duisburg-Essen
47048 Duisburg, Germany
kempken@inf.uni-due.de

Abstract

In this paper, we describe the software toolkit InterVerdiKom which implements a variety of reliable workload analysis techniques for queues with general or semi-Markovian arrival processes used in stochastic traffic modeling. We compute the workload at a network element in an open queuing network using transient and steady state analysis methods in interval arithmetic. We also present a modeling approach that employs a genetic programming optimization technique, providing an accurate model for real-life data in a compact state space, which is required for a successful verification. The resulting overflow probabilities of both the model and the empirical data are compared.

1. Introduction

The ongoing integration of services in Internet protocol (IP) networks requires an adequate support for quality of service (QoS) demands. At the same time, the number of traffic profiles that need to be modeled increases. Unfortunately, the characteristics of traffic in telecommunication networks are unpredictable from a number of viewpoints. For instance, the amount of transmission time and volume is unknown to network providers when transmission demands arise. Furthermore, sender and receiver cannot predict the amount of network resources available for the duration of the transmission. Randomly changing workload, routes and system parameters are both relevant and expected under normal operating conditions. In the end, failure events are unpredictable as well. However, the quality of the network services strongly depends on those properties.

Therefore, traffic modeling requires an appropriate basis for describing the data flow over time. We consider a stochastic approach employing random variables for modeling data arrivals at a given network element. On the one hand, one may consider the interarrival times between events like arrivals of packets, flows, connections or any other relevant units. This is the classical approach regarding queuing and service systems. On the other hand, time-slotted models can be applied that rely on the counting function of events (arrivals) in fixed time intervals [12, 4, 5].

Key characteristics of the arrival process are the distribution and the autocorrelation of the process. In order to model both adequately, we consider semi-Markovian stochastic processes (SMP) [7]. Furthermore, SMPs can be used to create both an interarrival / service time-based and a time-slotted model. Reliable and efficient analysis techniques for the resulting SMP/GI/1 queues are available; such models can be analyzed by matrix-analytical [11] or factorization techniques [12, 5, 14, 8, 18].

In this paper, we show how both modeling of realistic data and appropriate analysis techniques have to be combined to provide an integrated software toolkit for reliable traffic analysis: The recently developed software toolkit *InterVerdiKom* provides a versatile modeling approach for realistic data and also appropriate and reliable analysis techniques that compute the workload probabilities using factorization approaches. In previous work, we found that a successful verification of the workload analysis for semi-Markov models depends on the number of states of the model, which has to be kept sufficiently small [17].

Hence, we apply a modeling technique employing a genetic optimization approach to derive proper SMP models for real-life high definition (HD) video traffic in a compact state space in Section 2. Subsequently, we perform a verified workload analysis of the resulting queue models for both transient and steady states (Section 3) by means of interval arithmetic. Some general remarks on the toolkit are given in Section 4. In the end, we compare the results calculated for the overflow probability of a given network element for both the model and the original empirical data trace (Section 5).

2. Modeling of video traces

We introduce a discrete-time homogeneous semi-Markov process as a special case of a general semi-Markov process (cf. [10]). Such a process is given by a family of random variables $\{(S_t, \sigma_t)|t \in \mathbb{N}\}$ if

$$P(S_{t+1} = a, \sigma_{t+1} = j|S_k = a_k, \sigma_k = i_k, 1 \leq k \leq t) \\ = P(S_{t+1} = a, \sigma_{t+1} = j|\sigma_t = i_t)$$

holds for all $t \in N$ and $\{\sigma_t\}$ is a Markov chain. S_t represents the actual value at time t, and the distribution of S_{t+1} depends on the states σ_t and σ_{t+1} only. In the following, we restrict ourselves to the case of a finite Markov chain. If the underlying chain $\{\sigma_t\}$ has M states, the semi-Markov process is called SMP(M). A simplification can be achieved by using a special case SMP (SSMP): Here, the state-specific distribution of S depends not on the actual transition between states but on the current state only (cf. [7]). Please note that every SMP with M states can always be described as an SSMP with M^2 states. To do so, we identify each transition from state i to state j of the SMP as a state (i, j) of the SSMP (cf. [4]).

In the following, the entries of the state transition matrix P of the underlying Markov chain are given by p_{ij}, which is the probability of a transition of the Markov chain from state i to state j. The stationary probabilities for the system being in state i are given by p_i and can be computed by solving the following system of equations:

$$\sum_{i=1}^{M} p_i p_{ij} = p_j \text{ for } j = 1, \ldots, M; \sum_{i=1}^{M} p_i = 1 \qquad (1)$$

We consider a video trace of length N to be given as a sequence of amounts of data to be transmitted per time interval $R_j, j = 1, \ldots, N$, for instance, the number of bytes required for one frame or group of pictures. It is our intention to model two key characteristics of the video trace as accurately as possible: the distribution and the autocorrelation. The autocorrelation function of such a trace can be estimated by

$$\mathcal{A}_R(n) = \frac{1}{\sigma_R^2(N-n)} \sum_{j=1}^{N-n} (R_j - E(R))(R_{j+n} - E(R)). \qquad (2)$$

The autocorrelation function of an SSMP model is given by

$$\mathcal{A}_S(n) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} p_i E_i(S) p_{ij}^{(n)} E_j(S) - E^2(S)}{\sigma_S^2}. \qquad (3)$$

$E(S)$ is the expectation value of the whole process, while $E_i(S)$ denotes the expectation value if the underlying Markov chain is in state i. $p_{ij}^{(n)}$ is the n-step transition probability, that is, the probability that the process will change from state i to state j in n steps. In [7], we describe a way to efficiently evaluate the autocorrelation of an SSMP by writing it as an exponential sum $\mathcal{A}_S(n) = \sum_{i=2}^{M} \alpha_i \lambda_i^n$ with λ_i representing the eigenvalues of the transition matrix. The first eigenvalue λ_1 results to 1, since P is a stochastic matrix. The values α_i can be determined from some values of the autocorrelation function by solving a Vandermonde-type equation system.

We derive the SSMP parameters of a model for a given video trace – that is, the transition probabilities and the state-specific distribution – from an optimization process. We apply an evolutionary programming technique that does not derive the parameters from given data deterministically but instead provides an efficient test of new parameters and checks which fit best. To do so, a *population* is defined. Each member of this population (*individual*) represents a possible parameter configuration, and *fitness* values are computed for each one. The members with a high fitness level are reproduced, *mutated* and combined (*crossover*). They form the members of the next generation of the population. This depicts a simple but efficient optimization technique for a broad range of non-linear optimization problems, hence it may also be applied to the present task of parameter estimation. We found that this approach yields better results in a small state space than other deterministic modeling approaches [9]. Further details about genetic programming techniques can be found in, for instance, [20, 2].

First, we have to find a number of states M for the SSMP(M) model whose parameters are to be determined. In [7], we provided a verified method for the inclusion of the autocorrelation function. The bounds are given by interval coefficients of an exponential sum. We noticed that only a small number of coefficients are required to provide a verified enclosure of the autocorrelation function. Since the autocorrelation function of an SSMP(M) may be given as a sum of M exponential terms (with one term being set to zero due to the eigenvalue 1 of a stochastic matrix), we assume that only a correspondingly small number of states is required to model the autocorrelation function of the data trace. However, it remains an open question whether the accuracy of the SSMP model may be improved more efficiently by considering more states or by choosing the other parameters of a model with a given number of states more carefully.

We randomly assign each value of the empirical data trace to a state of the Markov chain of the SSMP. The states can be considered abstract and have no special meaning. If such an assignment is given for the whole sequence, all parameters of an SSMP model can be derived from it. The transition probabilities p_{ij} are computed by counting. Let the number of transitions from state i to state j be given by

n_{ij}; then the probabilities are estimated by:

$$p_{ij} := \frac{n_{ij}}{\sum_{k=1}^{M} n_{ik}} \tag{4}$$

Likewise, the state-dependent distributions can be estimated from the data values assigned to a particular state.

Our idea is to optimize the assignment iteratively so that the autocorrelation function of the original data $\mathcal{A}_R$ and the function of the SSMP derived from the assignment $\mathcal{A}_S$ match as closely as possible. To evaluate the quality of a particular assignment scheme and the derived SSMP respectively, we apply a fitness function:

$$g_R(S) := \frac{1}{\sum_{i=1}^{N}(\mathcal{A}_R(i) - \mathcal{A}_S(i))^2} \tag{5}$$

We start with a population of several random assignments and apply the following algorithm :

1. For each individual, construct an SSMP model with M states according to the given assignment scheme.
2. Compute the fitness of each individual.
3. Build up a new generation of individuals.
   (a) The best individual – that is, the one with the highest fitness value – is taken into the next generation unchanged.
   (b) A *mating pool* is constructed from a selection of the population. Each individual has a probability proportional to its fitness value for selection for the mating pool.
   (c) With given probabilities, some mutators are applied to the members of the mating pool (see below).
   (d) Genetic material is exchanged between random members of the mating pool (crossover, see below).
   (e) The changed and unchanged members of the mating pool form the next generation of the population.

The genetic operators (mutation and crossover) and their probability are important to the optimization process. In our case, we used four types of mutation operators:

- *Swap*. Two random values in the sequence are exchanged.
- *Reverse*. A randomly chosen, continuous part of the sequence is reversed.
- *Shuffle*. A randomly chosen, continuous part of the sequence is shuffled.
- *Block*. A randomly chosen, continuous part of the sequence is replaced by a block of identical random values.

In our experiments, we applied the so-called 2-point crossover: A randomly chosen continuous part of the genetic sequence is replaced by a sequence taken from another individual with the same length and relative position.

The main drawback of the evolutionary approach is that there is no rigorous criterion indicating that an optimal assignment has been reached. Hence, the optimization process could be extended almost infinitely. However, we abort this process when no improvement in terms of the highest fitness value of the current population could be achieved for a given number of iterations. As a result, we yield a matrix of transition probabilities that reflects the autocorrelation behavior of the empirical trace.

Next, we have to compute the discrete state-specific distributions for the arrival process. The SSMP approach allows us to adapt the autocorrelation and the state-specific distributions separately, since the autocorrelation depends only on the mean values per state. Hence, we have to preserve the mean value when performing a discretization of the GoP size distribution. For reasons of simplicity, we chose a uniform quantization of the values starting at 0.

We denote the number of values S_k to be discretized into d equidistant steps by K, and the difference between the discretization points as

$$\Delta = \frac{\max(\text{GoP-Size})}{d-1}.$$

We yield d points $s_l = l\Delta, l = 0, \ldots, d$. We denote the discretized variable by S_Δ. The probability $P(S_\Delta = s_l)$ is influenced by each value that differs less than Δ from s_l:

$$P(S_\Delta = s_l) = \sum_{k=1;|S_k - s_l|<\Delta}^{K} \frac{\Delta - |S_k - s_l|}{K\Delta} \tag{6}$$

3. Analysis

Using the modeling approach described, we are able to provide an adequate model for a single given video stream in terms of distribution and autocorrelation in a compact state space. Please note that multiple streams can be analyzed as well, since the superposition of multiple SMPs is also an SMP. However, the state space increases heavily: If, for instance, two SMPs with M_1 and M_2 states are to be multiplexed, the resulting SMP has $M_1 \times M_2$ states.

We now analyze the workload at a given network element such as a switch or router. In general, such a device accepts data from multiple input links and distributes them over a number of output links in the direction of their destination. The output line is chosen according to the particular

routing protocols. Usually, there is also a buffer available for each output link that stores all outgoing packets in case of collisions or temporary overload.

Thus, the usual router or switch model includes an output link with a given forwarding capacity S, a buffer of size B and an input process describing all packets that arrive at the switch and are transmitted over the output link.

In our case, we describe the input process using an SSMP and assume the service capacity S to be constant over time. We analyze the system in time slots of equal length L and consider the amount of data arriving at and leaving the switch per slot. For instance, the maximum amount of data transmitted from the switch is given by $S \times L$.

We consider a time-slotted queueing system. In the following text, the amount of data arriving in the time slot beginning at t is given by A_t. We describe the arrival process A by an SSMP and determine its parameters according to the method described in the previous section for fitting them to given empirical data. The difference variable U describing the workload change per time slot is therefore given by $U_t = A_t - S$. Our intention is to compute probabilities for certain workload levels, that is, the waiting time for newly arriving data packets and also the amount of data stored in the buffer of the network element.

3.1. Wiener-Hopf factorization

The Wiener-Hopf factorization approach provides a means to compute the steady state workload probabilities. This approach is efficient in terms of memory requirements and computation complexity compared to matrix-analytical methods. A verification of the results computed is feasible by Brouwer's fixed point theorem, as is explained below.

We consider the difference distribution $u_{ij}(k)$ given in matrix notation:

$$u_{ij}(k) = P(U_{t+1} = k, \sigma_{t+1} = j | \sigma_t = i), i, j \in Z, k \in \mathbb{Z} \tag{7}$$

with the states of the embedded Markov chain $\sigma \in Z := \{1, \ldots, M\}$. We assume that the distribution of A_n is limited by g and S_n by h correspondingly: $A_n \in \{0, \ldots, g\}, S_n \in \{0, \ldots, h\}$.

The basic idea of the Wiener-Hopf factorization, which is presented in detail e.g. by Hasslinger [5], is to consider two phase distributions. If no arrivals occur for a period of time, this period is considered as an "idle phase". Correspondingly, a "phase of workload level k" describes a period of time with a workload always higher than or equal to k. We consider the idle phase distribution $l_{ij}(k)$ representing the probability that an idle phase lasts for k slots and starts in state i and ends in state j, and the distribution $v_{ij}(k)$ representing the probability that a phase with workload level k, that is, the waiting time of a current arrival, and initial state j occurs in a phase with initial state i. We write these two distributions in matrix notation:

$$\mathbf{l}(k) = (l_{ij}(k)); \mathbf{v}(k) = (v_{ij}(k)); i, j \in Z, k \in \mathbb{N}_0$$

There is a relation between these two distributions such that a phase of workload level $\nu + \mu$ and initial state i is followed by a phase of level μ and initial state j with a probability of $l_{ij}(\mu)$. This leads to the following equations (cf. [5]):

$$\begin{aligned}
\mathbf{v}(\nu) &= \mathbf{u}(\nu) + \sum_{\mu=1}^{\min(h-\nu,g)} \mathbf{v}(\nu+\mu)\mathbf{l}(\mu), \\
&\quad \nu = h, \ldots, 0; \\
\mathbf{l}(\nu) &= \mathbf{u}(-\nu) + \sum_{\mu=0}^{\min(g-\nu,h)} \mathbf{v}(\mu)\mathbf{l}(\mu+\nu), \\
&\quad \nu = g, \ldots, 1; \\
\mathbf{v}(\nu) &= \mathbf{0}, \nu > h, \\
\mathbf{l}(\nu) &= \mathbf{0}, \nu > g
\end{aligned}$$

We may compute these distributions with the Grassmann-Jain algorithm [3]. We start with an approximation for

$$l_{ij}^{(0)}(k) = \frac{u_{ij}(k)}{\sum_{n=-g}^{-1} u_{ij}(n)}; i, j \in Z.$$

Now we compute an approximation for $v_{ij}^{(0)}(k)$ from the results above. We continue to compute the distributions by alternating between the computation of $\mathbf{l}(k)$ and $\mathbf{v}(k)$ until we reach convergence.

The convergence of this approach has only been proven for the GI/G/1 case. Here, we apply a computer-based proof using interval arithmetic. We enclose the results yielded by the floating-point approximation in tight intervals (e.g. diameter 10^{-14}) and perform a step of the iteration using interval variants of the corresponding equations. If all of the newly computed values $[v_{ij}(k)]$ are contained in the old intervals, Brouwer's fixed point theorem guarantees that the limit value of the iteration is included in the new intervals.

From the distributions $v_{ij}(k)$ and $l_{ij}(k)$, we are able to derive the workload probabilities of the observed queue. We outline the method below, further details are given in [1]. The utilization of interval arithmetic in our implementation guarantees verified enclosures of the results.

We compute the steady state distribution of the embedded Markov chain in idle phases l_j and the distribution of the length of the idle phases $l(k)$ from the following system

of equations:

$$l_j := \sum_{i=1}^{M} l_i l_{ij}, j \in Z, ; \sum_{j=1}^{M} l_j = 1; \quad (8)$$

$$l(k) := \sum_{i,j=1}^{M} l_i l_{ij}(k), k \in \mathbb{N}_0 \quad (9)$$

Let $\mathcal{W}_i(z) = \sum_{k=0}^{\infty} w_i(k) z^k$ denote the generating function of the probability $w_i(z)$ that the workload of the queue is z in a busy period with initial state i, N_i the number of demands served per busy period and $E(N_i)$ the corresponding expectation value. Then the following relations hold:

$$E(N_i) = 1 + \sum_{j=1}^{M} \sum_{n=0}^{h} v_{ij}(n) E(N_j); \quad (10)$$

$$\mathcal{W}_i(z) E(N_i) = 1 + \sum_{j=1}^{M} \sum_{n=0}^{h} v_{ij}(n) z^n \mathcal{W}_j(z) E(N_j). \quad (11)$$

This leads to

$$\mathcal{W}(z) = \frac{\sum_{i=1}^{M} l_i E(N_i) \mathcal{W}_i(z)}{\sum_{i=1}^{M} l_i E(N_i)} \quad (12)$$

providing the generating function for the workload probabilities at an arbitrary arrival time.

3.2. Transient analysis

Transient analysis of stochastic systems is a basic method related to both steady state analysis and simulation. It describes the evolution of the system from a given starting point over time and also its long-term development. If the system is ergodic, a convergence to a steady state is observed.

In comparison to simulation, the transient analysis provides directly complete distribution functions at embedded time points for the states of the system being observed. Simulation follows randomly chosen paths of the system development, therefore the results are subject to statistical deviations within confidence levels.

We consider the workload of a time-slotted semi-Markovian server with M states. Let the current state of the system at the beginning of time slot t be given by σ_t and the current workload by w_t. In this case,

$$W_t = (w_t, \sigma_t) \in \mathbb{N}_0 \times Z := \{1, \ldots, M\} \quad (13)$$

provides all necessary information on the current state of the system relevant to the future development of its workload. Z represents the set of states of the Markov chain.

As a matter of fact, changes in the current workload level will occur according to the particular arrival and service distributions. A_t and S_t are random variables that count the number of arrivals and service events during the time slot t. We assume that the service occurs at the end of the time slot. In this case, $U_t = A_t - S_t$ describes the increase or decrease of the workload during time slot t. The new workload is given by Lindley's equation [13]:

$$w_{t+1} = \max(w_t + U_t, 0) = \max(w_t + A_t - S_t, 0) \quad (14)$$

with $w_0 = 0$.

In our case, we may assume that A_t and S_t are bounded:

$$0 \leq A_t \leq g, 0 \leq S_t \leq h, -h \leq U_t \leq g$$

We describe the arrival and service distributions according to a semi-Markov process:

$$\begin{aligned} &a_i(k) = P(A_t = k | \sigma_t = i) \text{ for } 0 \leq k \leq h, i \in Z \\ &s(k) = P(S_t = k) \text{ for } 0 \leq k \leq g \\ \Rightarrow &u_i(k) = P(A_t - S_t = k | \sigma_t = i) \text{ for } -g \leq k \leq h \end{aligned}$$

$a_i(k)$, which is the state-specific arrival distribution, is defined by the probability of k arrivals in a time slot if the system is in state i at the beginning of the slot. We assume the service distribution to be independent of the state of the system. From one time slot to another, the system may change the state of the Markov process according to the transition matrix $P = (p_{ij})$, $i, j \in Z$:

$$p_{ij} = P(\sigma_{t+1} = j | \sigma_t = i)$$

According to the previously defined behavior of a semi-Markovian server, we obtain the transition probabilities of the state W_t to W_{t+1}:

$$P(W_{t+1} = (l, j) | W_t = (k, i)) = u_i(l - k) p_{ij} \quad (15)$$

We assume the workload to be limited to a maximum of N. In practice, network elements that are to be modeled also have only a limited buffer capacity. Any additional data that arrives if the buffer is full is dropped. Therefore, we introduce two additional equations for the boundary states of a workload level of 0 and N:

$$P(W_{t+1} = (0, j) | W_t = (k, i)) = \sum_{n=-g}^{-k} u_i(n) p_{ij} \quad (16)$$

$$P(W_{t+1} = (N, j) | W_t = (k, i)) = \sum_{n=N-k}^{h} u_i(n) p_{ij} \quad (17)$$

Hence, we may iteratively compute the state probabilities of the queue. As a stability condition, we require the mean service capacity to be larger than the mean arrival rate:

$$E(A) < E(S) \Leftrightarrow E(U) < 0 \quad (18)$$

In this case, the transient distribution of the workload converges to the steady state solution, as has already been shown by Lindley [13]. This convergence implies that for every $\epsilon > 0$ there is a point n such that

$$|w_n(k) - w(k)| < \epsilon.$$

This is true for every $0 \leq k \leq N$ with $w_n(k)$ representing the probability of workload k at time n computed by transient analysis and $w(k)$ being the steady state probability of a workload level k, which can be determined, for instance, using Wiener-Hopf factorization as presented above. If the parameter N chosen is sufficiently large, we may neglect the additional probabilities of higher workloads.

Using interval arithmetic as a reliable technique, we are able to do a "worst-case" analysis for given values ϵ (cf. [7]): We consider the maximum difference between the outer bounds of the intervals computed by transient and steady state analysis. If this difference is less than ϵ, we know that at most n iterations are required. A more precise implementation of either the transient or steady state analysis method resulting in smaller rounding errors may yield a smaller number n that is a better approximation of the exact number of iterations required.

When we determine the workload distribution using transient analysis, we assume the queue to be empty at the beginning. In contrast, the steady state analysis provides results for the system being at a "normal" workload level. The number n we gained from the comparison of transient and steady-state analysis may be considered as an indicator for the length of a "warm-up" phase of the queue – that is, the time required to reach a level of workload normal for the amount of data to be processed.

4. InterVerdiKom

InterVerdiKom provides the user with a means to apply the analysis techniques presented to queuing systems for both steady state and transient analysis. Altogether, these methods include

- reliable steady state workload analysis based on factorization of the characteristic polynomial of an SMP/G/1 or GI/G/1 queue,
- Wiener-Hopf factorization with verified workload distribution results for GI/G/1 and SMP/G/1 queues,
- reliable computation of the transient and steady state queue size probability distribution for GI/G/1 queues,
- verified transient analysis of GI/G/1 and SMP/G/1 queues in terms of workload probabilities in a compact state space.

The methods mentioned are implemented in C++ with the additional C-XSC library for extended scientific computing, providing interval arithmetic functionality. Through use of the Qt library by Trolltech, the graphical user interface is platform-independent. However, due to the use of the library C-XSC the InterVerdiKom tool functions only on supported platforms (currently UNIX / Linux and Macintosh). A Microsoft Windows version may be technically possible using the Cygwin environment. Integration with other analytical tools is provided by text-based files, as input and output data is given in standard text files.

5. Examples

As an example, we consider the overflow probabilities at a given network element such as a switch or router. We estimate the model parameters for a high definition (HD) video stream encoded in H.264 / AVC format. Please note that the techniques presented may be applied to any other variable bit rate format as well.

We consider the "Horizon" and "From Mars to China" (Mars) data traces provided by van der Auwera and Reisslein [19, 16], which are publicly available for evaluation purposes on their web site[1]. Some statistical details about the traces are given in Table 1.

Characteristic	"Horizon"	"Mars"
Length (GoP)	4,099	4,310
Min size (Byte)	2,624	10,832
Max size (Byte)	2,918,216	8,803,048
Mean size (Byte)	613,777	1,939,697
Standard dev. σ (Byte)	210,718	1,110,315

Table 1. Trace characteristics

Beginning with the video trace files, we first combine the individual frame size values to a series of GoP sizes. We consider a group of pictures as the basic transmission unit for several reasons: On the one hand, it is more accurate to model a sequence of GoP sizes instead of frames because, due to the coding scheme used, MPEG frame sizes show a periodic behavior. This periodic behavior requires a high number of SMP states to be modeled adequately. However, this characteristic is easy to reconstruct from given GoP sizes [15]. On the other hand, the particular frames in MPEG streams are not transmitted in the same order as they are played back in the video sequence, but re-sorted in order to make the use of bi-directional coding feasible.

The sequence of GoP sizes is modeled as an SSMP according to the approach presented in Section 2: The parameters and state-specific distributions are derived from

[1] http://trace.eas.asu.edu/

Clip	Horizon		Mars	
Capacity	InterVerdiKom	Empirical trace	InterVerdiKom	Empirical trace
$S = E(G) + 1\sigma(G)$	0.1571	0.2189	0.2991	0.2121
$S = E(G) + 2\sigma(G)$	0.0364	0.0478	0.0413	0.0637
$S = E(G) + 3\sigma(G)$	0.0168	0.0242	0.0091	0.0141
$S = E(G) + 4\sigma(G)$	0.0107	0.0173	0.0017	0.0021

Table 2. Overflow probabilities $P(W > 0) = 1 - P(W = 0)$

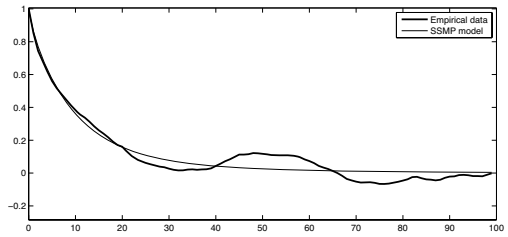


Figure 1. Autocorrelation function, "Horizon" clip

the empirical data trace using a genetic programming algorithm. We see that the autocorrelation of the original data trace and the SSMP model match quite well (see Figures 2, 1). Since the state-specific distributions are directly computed from the original trace, they are correct up to a discretization error. To make the succeeding calculations feasible, we therefore have to make a compromise between accuracy and computation complexity. In the examples presented, we consider the size distributions in steps of $50,000$ bytes.

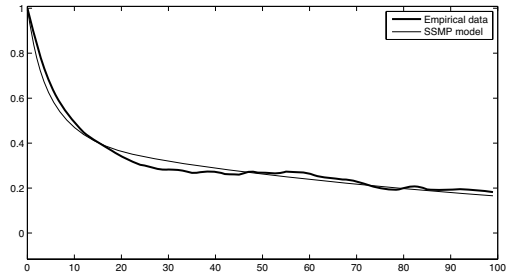


Figure 2. Autocorrelation function, "Mars" clip

We may now analyze the overflow probabilities – that is, the probability of a workload higher than zero – for various output capacities for both the model and the original data trace. For the latter, we compute the workloads

$$W_n = \max(0, W_{n-1} + G_n - S) \tag{19}$$

with G_n being the GoP sizes from the empirical trace and S the fixed service capacity, which is assumed to be

$$S = E(G) + k\sigma(G) \tag{20}$$

with parameter $k \in \mathbb{N}$.

The distribution of the workloads is also discretized to match the model steps. In the two examples, we consider SSMP models with 5 states. As mentioned, the number of states is a parameter to the optimization process. In the present case, we found that the resulting models are accurate enough to reflect the autocorrelation behavior and also small enough to be analyzed with verification techniques. The workload distribution of the SSMP model is computed according to the Wiener-Hopf method presented in Section 3.1. The utilization of this approach yields tight intervals for the workload distribution, as can be seen in Table 3 for one exemplary computation. We give a comparison of the overflow probabilities in Table 2, which are given by the first digits only for the sake of clarity.

The results show that, considering the fact that only a small number of states is used in the model, the outcoming overflow probabilities give an adequate approximation of the values of the empirical trace.

$P(W = 0)$	$9.98313517 8^{6183220}_{5639799} \cdot 10^{-1}$
$P(W = 50,000)$	$1.7171549896^{538977}_{455072} \cdot 10^{-6}$
$P(W = 100,000)$	$1.367624073 19^{93988}_{27163} \cdot 10^{-6}$
$P(W = 150,000)$	$5.79858051 12^{780075}_{481317} \cdot 10^{-5}$
$P(W = 200,000)$	$1.161578714 77^{68189}_{08304} \cdot 10^{-4}$
$P(W = 250,000)$	$5.056328517 4^{409542}_{149103} \cdot 10^{-5}$

Table 3. Workload probabilities for SSMP(5) model of "Mars" clip, $S = E(G) + 4\sigma(G)$

The transient analysis method presented in Section 3.2 allows us to perform a verified "worst-case" analysis of the

required number of arrivals n so that the workload distribution computed by iteration and steady state analysis match for a given tolerance ϵ. To do so, we determine the maximum difference between the outer bounds of the intervals derived by transient and steady state analysis. After n iterations, this difference is less than ϵ.

We consider an example with 4 states. The transition matrix is given by

$$P = (p_{ij}) = \begin{pmatrix} 0.7 & 0.3 & 0.0 & 0.0 \\ 0.2 & 0.7 & 0.1 & 0.0 \\ 0.0 & 0.1 & 0.8 & 0.1 \\ 0.0 & 0.0 & 0.3 & 0.7 \end{pmatrix}$$

and the state-dependent arrival distributions A_i are:

$$\begin{array}{llll} P(A_1 = 71) = 0.1 & \dots & P(A_1 = 80) = 0.1 \\ P(A_2 = 81) = 0.05 & \dots & P(A_2 = 100) = 0.05 \\ P(A_3 = 91) = 0.05 & \dots & P(A_3 = 110) = 0.05 \\ P(A_4 = 101) = 0.05 & \dots & P(A_4 = 120) = 0.05 \end{array}$$

The service capacity is assumed to be either at 100 units or 105 units per time slot, both with a probability of 0.5. The resulting numbers n for some values ϵ are displayed in Table 4.

$\epsilon = 10^{(\cdot)}$	-1	-2	-3	-4	-5	-6	-7
n	21	46	79	123	175	232	292

Table 4. Required number of iterations for different values ϵ

6. Conclusion and further work

In this paper, we have presented reliable and accurate analysis techniques for semi-Markovian queuing systems implemented in the toolkit *InterVerdiKom*.

We presented a new approach to derive the parameters of an SSMP queuing system that employs a genetic programming technique. Using this approach, we are able to form SSMP models with a small number of states that can reflect the autocorrelation behavior of an empirical trace. Since the complexity of the SSMP models has been a problem in previous work ([6, 17]) preventing the outcome of verified results in some cases, the technique proposed in this paper allows the verified analysis of realistic video traces with a more complex behavior.

The steady state workload distribution of an SMP/G/1 queue can be determined by Wiener-Hopf factorization. The usage of interval arithmetic provides reliable outcomes and allows verification of the numerical results necessary for quality of service purposes. Furthermore, we presented a means for analyzing transient states of the queue. Here, interval arithmetic provides round-off error control, thus providing a worst-case analysis technique for the convergence behavior. The duration of the "warm-up" phase of an empty queuing system can also be estimated using both transient and steady state analysis.

Our toolkit *InterVerdiKom* provides implementations of these methods and their use through a convenient graphical user interface. We also included the modeling approach presented in here to provide an integrated set of video traffic modeling and analysis methods.

Further work is planned on the analysis of arrival distributions in terms of the convolutions and superpositions of several distributions and also the aggregation of multiple semi-Markov processes. We also want to expand our tool to allow various forwarding strategies like reservation, prioritization, fair bandwidth sharing and packet drop policies. Also, we intend to enhance InterVerdiKom to include analysis techniques for queuing networks.

7. Acknowledgments

This research was carried out in an ongoing project funded by the German Research Council (DFG).

References

[1] D. Fausten and G. Haßlinger. Verified numerical analysis of the performance of switching systems in telecommunication. In *Numerical Software with Result Verification*, pp. 209–228, 2004.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.

[3] W. K. Grassmann and J. L. Jain. Numerical solutions of the waiting time distribution and idle time distribution of the arithmetic GI/G/1 queue. *Operations Research*, 37(1):141–150, 1989.

[4] G. Haßlinger. Semi-Markovian modelling and performance analysis of variable rate traffic in ATM networks. *Telecommunication Systems*, 7:281 – 298, 1997.

[5] G. Haßlinger. Waiting times, busy periods and output models of a server analyzed via Wiener-Hopf factorization. *Performance Evaluation*, 40:3–26, 2000.

[6] G. Hasslinger and D. Fausten. Verified performance analysis of real time video multiplexing. In *Performance and Control of Next-Generation Communications Networks*, 2003.

[7] S. Kempken and W. Luther. Verified methods in stochastic traffic modeling. In *Reliable Implementation of Real Number Algorithms: Theory and Practice*, LNCS. Springer, 2006. (to appear)

[8] S. Kempken, W. Luther, and G. Haßlinger. A tool for verified analysis of transient and steady states of queues. In *Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '06)*, 2006. (electronic publication)

[9] S. Kempken and W. Luther. Modeling of H.264 high definition video traffic using discrete-time semi-Markov processes. In *Proceedings of the 20th International Teletraffic Congress*, 2007. (to appear)

[10] L. Kleinrock. *Queueing Systems*, volume 1/2. Wiley, 1975.

[11] G. Latouche and V. Ramasvami. *Introduction to matrix analytic methods in stochastic modeling*. ASA-SIAM, 1999.

[12] S. Li. A general solution technique for discrete queueing analysis of multi-media traffic on ATM. *IEEE Transactions on Communication*, 39:1115–1132, 1991.

[13] D. Lindley. The theory of queues with a single server. In *Proc. Cambridge Philos. Soc.*, volume 48, pp. 277 – 289, 1952.

[14] W. Luther and S. Kempken. Reliable computation of workload distributions using semi-Markov processes. In *Proceedings of the 13th International Conference on Analytical and Stochastic Modelling Techniques and Applications*, pp. 111 – 116, 2006.

[15] O. Rose. Simple and efficient models for variable bit rate MPEG video traffic. *Performance Evaluation*, 30:69 – 85, 1997.

[16] P. Seeling, M. Reisslein, and B. Kulapala. Network performance evaluation with frame size and quality traces of single-layer and two-layer video: A tutorial. *IEEE Communications Surveys and Tutorials*, 6(3):58 – 78, 2004.

[17] D. Traczinski. *Faktorisierungslösungen für die Workload in Bediensystemen mit Ergebnisverifikation*. PhD thesis, Universität Duisburg-Essen, 2005.

[18] D. Traczinski, W. Luther, and G. Haßlinger. Polynomial factorization for servers with semi-Markovian workload: Performance and numerical aspects of a verified solution technique. *Stochastic Models*, 21:643–668, 2005.

[19] G. van der Auwera, P. David, and M. Reisslein. Bit rate-variability of H.264/AVC FREXt. Technical report, Arizona State University, 2006.

[20] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

[21] H. Takagi and D.-A. Wu. Multiserver queue with semi-Markovian batch arrivals with application to the MPEG frame sequence. *Proc. Internet Performance and Control of Network Systems* SPIE 4865, pp. 178 – 189, 2002.

Efficient 16-bit Floating-Point Interval Processor for Embedded Systems and Applications

Stéphane Piskorski
Laboratoire de Recherche en Informatique
CNRS – Univ Paris-Sud
F-91405 Orsay Cedex, France

Lionel Lacassagne
Institut d'Electronique Fondamentale
CNRS – Univ Paris-Sud
F-91405 Orsay Cedex, France

Michel Kieffer
Laboratoire des Signaux et Systèmes
CNRS – Supélec – Univ Paris-Sud
F-91192 Gif-sur-Yvette, France

Daniel Etiemble
Laboratoire de Recherche en Informatique
CNRS – Univ Paris-Sud
F-91405 Orsay Cedex, France

Abstract

This paper presents the implementation of a 16-bit interval floating-point unit on a soft-core processor to allow interval computations for embedded systems. The distributed localization of a source using a network of sensors is presented to compare the performance of the proposed processor to that obtained with a general-purpose processor.

1 Introduction

In the last ten years, interval techniques [21, 22] have allowed original solutions for many problems in engineering to be proposed, see, *e.g.*, [12]. One of the main features of interval techniques is their ability to provide *guaranteed* results, *i.e.*, with a verified accuracy or which are numerically *proved*. Consider for example, a bounded-error parameter estimation problem: the value of some parameter vector has to be estimated from measured data using a given model structure and bounded measurement errors. In such a context, one may obtain a set which can be proved to contain all values of the parameter vector that are consistent with the model structure, the measured data, and the hypotheses on the noise. Nevertheless, the application of interval techniques in embedded real-time applications is far less developed. The lack of efficient interval hardware support may be a reason for this slower development.

Hardware implementations of interval arithmetic have been mentioned twenty years ago in [18]. Extension of existing hardware platforms have been proposed, *e.g.*, in [28] and [17]. Nevertheless, chip builders were not yet convinced of the usefulness of performing specific adaptation of chips to implement interval analysis. This is why interval analysis is mainly performed by software implementations on general-purpose processors. Interval computations are however quite inefficiently performed on such processors, since the recurrent rounding mode switchings required by interval computations results in recurrent flushes of the processor pipeline [10]. This specific problem led people to study and design dedicated floating-point units (FPU) well suited to double rounding modes (towards $-\infty$ and towards $+\infty$) [17]. Moreover, in many applications, 32-bit FPU are oversized. Measurements, corrupted by errors, do not require to be processed with such an accuracy and in many cases, smaller FPU with reduced precision may fit the application constraints and provide a satisfying accuracy. Thus, for example, 16-bit floating-point computations is an efficient way to tackle both accuracy and dynamic problems encountered in signal and image processing [19], for filtering and convolution-based algorithms.

This paper introduces 16-bit floating-point arithmetic adapted to interval computations. The main idea is inspired by [17], which proposed to implement two 32-bit FPU on the 64-bit FPU of a general-purpose processor. Here, similarly, noticing that a 16-bit FPU is smaller than a 32-bit FPU, two 16-bit FPU (managing the two rounding modes required for interval computations) are shown not being much bigger than a single 32-bit FPU. The main advantage is that no rounding mode switching is required, preventing them from

flushing the processor pipeline. The implementation of such a 16-bit FPU is performed on the FPGA based NIOS-II soft processor [4, 7], which allows instructions to be added to its instruction set. Customizable processors represent an opportunity to propose efficient and low-cost on-chip interval applications which may be used in embedded applications.

To compare the performance of 16 and 32-bit FPUs, an example of source localization using a network of acoustic or electromagnetic sensors is considered. In such network of sensors, power consumption and computational complexity are strong constraints when one is concerned with the increase of operability and autonomy [1]. Distributed interval constraint propagation [14] has been proposed as an efficient and low-complexity solution for source localization using a network of wireless sensors.

Section 2 recalls the distributed source localization problem and sketches the solutions based on interval analysis. In Section 3, the architecture of the 16-bit FPU is presented. Attention is paid to accuracy and dynamic range. Results provided by a 32-bit FPU are compared to those obtained with two 16-bit FPU on realistic simulated data. Section 4 describes the hardware implementation on the three targeted architectures (Pentium4, Pentium 4-M, and NIOS-II) and provides benchmarks for execution time and energy consumption.

2 Source localization with a network of sensors

Localizing a source emitting an acoustic or electromagnetic wave using a set of distributed sensors has received a growing attention in recent years, see, *e.g.*, [27, 26]. The localization technique used depends on the type of information available to the sensor nodes. Time of arrival (TOA), time difference of arrival (TDOA), and angle of arrival (AOA) usually provide the best results [24]. Nevertheless, these quantities are rather difficult to obtain, as they require a good synchronization between timers (for TOA), collaboration between neighboring sensors (for TDOA), or multiple antennas (for AOA). Contrary to TOA, TDOA, or AOA data, readings of signal strength (RSS) at a given sensor are easily obtained, as they only require low-cost measurement devices or are already available, as in IEEE 802.11 wireless networks, where these data are provided by the MAC layer [26].

This paper focuses on source localization from RSS data using a distributed localization technique. Distributed means that each sensor processes its own measurements and exchanges partially processed data with its neighbors. This approach has the advantage of being more robust to sensor failures than centralized localization, where all measurements are processed by a single unit. Distributed approaches have been employed, *e.g.*, in [25], where a distributed version of nonlinear least squares has been presented. When badly initialized, this approach suffers from convergence problems, as illustrated in [8], which advocates projection on convex sets. However, this requires an accurate knowledge of the source signal strength and of the path loss exponent. In [14], a centralized and distributed localization technique based on interval analysis has been proposed. All measurement noises are assumed to be bounded with known bounds. The problem is then cast into a set-inversion problem, which is solved in a centralized fashion using SIVIA [13] and in a distributed manner using interval constraint propagation [12].

The model, hypotheses, and main results of [14] are briefly recalled here, before introducing the implementation of the localization algorithm on a dedicated processor.

2.1 Source localization from bounded-error data

The single source localization problem illustrated in Figure 1 is considered. The unknown location of the

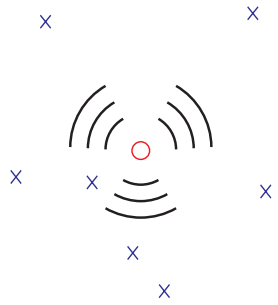


Figure 1. Source (o, unknown location θ) and sensors (x, known location $\mathbf{r}_\ell$)

source is denoted by $\boldsymbol{\theta} \in \mathbb{R}^2$; $\mathbf{r}_\ell \in \mathbb{R}^2$, $\ell = 1 \dots L$ represents the known location of the ℓ-th sensor. The RSS by sensor ℓ is denoted by y_ℓ. Using the Okumura-Hata model [23], describing the mean power received by the ℓ-th sensor when a source at a distance $|\mathbf{r}_\ell - \boldsymbol{\theta}|$ emits a wave with power A (measured at a distance of 1 m), one may write

$$y_\ell = y_{\mathrm{m}}\left(\boldsymbol{\theta}, A, n_{\mathrm{p}}, \ell\right) w_\ell, \ell = 1 \dots L \qquad (1)$$

with

$$y_{\mathrm{m}}(\boldsymbol{\theta}, A, n_{\mathrm{p}}, \ell) = \frac{A}{|\mathbf{r}_\ell - \boldsymbol{\theta}|^{n_{\mathrm{p}}}}. \quad (2)$$

The noise $w_\ell \in [w]$ is multiplicative, as it is additive in the log domain. In (2), n_{p} is the path-loss exponent. Both A and n_{p} are assumed unknown. The parameter vector to be estimated is thus $\mathbf{p} = (\boldsymbol{\theta}^{\mathrm{T}}, A, n_{\mathrm{p}})^{\mathrm{T}}$ and $y_{\mathrm{m}}(\boldsymbol{\theta}, A, n_{\mathrm{p}}, \ell)$ will be written as $y_{\mathrm{m}}(\mathbf{p}, \ell)$.

The aim here is to characterize the *set* $\mathbb{P} \subset [\mathbf{p}]_0$ of all parameter vectors that are *consistent* with the measurements, the RSS model (2), and the noise bounds. The initial search box $[\mathbf{p}]_0$ is assumed to contain the actual parameter value $\mathbf{p}^*$. $\mathbb{P}$ may then be defined as follows

$$\mathbb{P} = \{\mathbf{p} \in [\mathbf{p}]_0 \mid y_{\mathrm{m}}(\mathbf{p}, \ell) \in [y_\ell], \ell = 1 \ldots L\}, \quad (3)$$

where $[y_\ell] = y_\ell / [w]$ is assumed to contain the actual noise-free RSS by sensor ℓ.

The characterization of $\mathbb{P}$ when all measurements y_ℓ are available at a given location of the sensors network is a classical set-inversion problem which may be solved using set-inversion techniques, like SIVIA [13]. SIVIA provides two *subpavings* (union of non-overlapping boxes) $\underline{\mathbb{P}}$ and $\overline{\mathbb{P}}$, such that

$$\underline{\mathbb{P}} \subset \mathbb{P} \subset \overline{\mathbb{P}}. \quad (4)$$

The accuracy of the description of $\mathbb{P}$ is controlled by a parameter ε, which determines the width of the smallest box in $\underline{\mathbb{P}}$ and $\overline{\mathbb{P}}$.

This approach may be easily extended in a distributed context: each sensor ℓ may evaluate two subpavings $\underline{\mathbb{P}}_\ell$ and $\overline{\mathbb{P}}_\ell$, guaranteed to enclose the set

$$\mathbb{P}_\ell = \{\mathbf{p} \in [\mathbf{p}]_0 \mid y_{\mathrm{m}}(\mathbf{p}, \ell) \in [y_\ell]\}, \quad (5)$$

of all values of $\mathbf{p}$ consistent with the ℓ-th measurement. The solution set $\mathbb{P}$ in (3) is then obtained by intersecting the sets $\mathbb{P}_\ell$, $\ell = 1 \ldots L$

$$\mathbb{P} = \bigcap_{\ell=1}^{L} \mathbb{P}_\ell. \quad (6)$$

Performing this intersection is far from being trivial. This requires the transmission of the solution sets $\mathbb{P}_\ell$ between sensors. Such exchanges require many communications, a highly energy consuming task, especially when the desired accuracy (determined by ε) is high, resulting in a large number of boxes stored in each subpaving. The next section describes an alternative approach based on interval constraint propagation (ICP) [12].

2.2 Distributed localization: interval constraint propagation

Here, contrary to classical constraint satisfaction problems, the variables and constraints are *distributed* over the sensor network, [5, 2]. Nevertheless, ICP may still be used at each individual sensor.

At a sensor ℓ, the variables are y_ℓ, $\boldsymbol{\theta}$, A, and n_{p}, their domains are $[y_\ell]$, measured at the sensor, and $[\boldsymbol{\theta}]$, $[A]$, and $[n_{\mathrm{p}}]$, obtained from its neighbors. The variables must satisfy the constraint provided by the RSS model (2), thus

$$y_\ell - \frac{A}{|\mathbf{r}_\ell - \boldsymbol{\theta}|^{n_{\mathrm{p}}}} = 0. \quad (7)$$

From (7), the *contracted domains* [12] may be written as

$$\begin{aligned}
[y'_\ell] &= [y_\ell] \cap \frac{[A]}{|\mathbf{r}_\ell - [\boldsymbol{\theta}]|^{[n_{\mathrm{p}}]}}, \\
[A'] &= [A] \cap [y'_\ell]\, |\mathbf{r}_\ell - [\boldsymbol{\theta}]|^{[n_{\mathrm{p}}]}, \\
[n'_{\mathrm{p}}] &= [n_{\mathrm{p}}] \cap (\log([A']) - \log([y'_\ell])) / \log(|\mathbf{r}_\ell - [\boldsymbol{\theta}]|), \\
[\theta'_1] &= [\theta_1] \cap \left(r_{\ell,1} \pm \sqrt{([A'] / [y'_\ell])^{2/[n'_{\mathrm{p}}]} - (r_{\ell,2} - [\theta_2])^2} \right), \\
[\theta'_2] &= [\theta_2] \cap \left(r_{\ell,2} \pm \sqrt{([A'] / [y'_\ell])^{2/[n'_{\mathrm{p}}]} - (r_{\ell,1} - [\theta_1])^2} \right).
\end{aligned}$$

In the last two update equations, the set intersecting $[\theta_1]$ and $[\theta_2]$ may consist of two disconnected intervals. In this case, the smallest interval containing the result is evaluated. For each sensor, ICP provides thus a box which is guaranteed to contain the set $\mathbb{P}$ defined by (3).

Using [11], it can be shown that the contraction is optimal with respect to the information available at the ℓ-th sensor. However, when considering all constraints simultaneously, the optimality conditions no longer hold. Cycling through the sensor network, as in [8, 25] improves the estimation. More details about the optimization of sensor communications may be found in [5].

In this distributed approach, a single box has to be transmitted from one sensor to its neighbors, which is much less energy consuming than transmitting a subpaving. To perform all computations at a given sensor, basic interval arithmetic operations have to be implemented ($+, -, \times$, and $/$), the intersection between intervals, and the square root and square of an interval. Elementary functions $\ln(\cdot)$ and $\exp(\cdot)$ have also to be provided. The next section shows how interval constraint propagation may be implemented on a dedicated FPGA using 16-bit floating-point numbers.

3 Why 16-bit floating-point numbers?

On one hand, for some applications in image and signal processing (*e.g.*, filtering), 32-bit floating-point (FP) numbers are oversized. On the other hand, fixed-point arithmetic may be inaccurate (*e.g.,* when considering recursive filtering, motion estimation, image stabilization). Moreover, for embedded applications, a 32-bit FPU is larger than a 16-bit FPU and thus consumes more power. As interval computation requires rounding mode switching, designing a customized 16-bit interval FPU is a way to tackle both problems together, as will be seen with the Altera NIOS-II soft processor [4].

3.1 F16 format

Some years ago, a 16-bit FP format called *half* has been introduced in the OpenEXR format [6] and in the Cg language [20] defined by NVIDIA. It is currently used in some NVIDIA and ATI graphical processing units (GPU) for transformation and lightning (3D computations) and has been introduced in the draft of the IEEE 754r standard [9].

The *half* format is justified by ILM (Industrial Light and Magic of George Lucas), which developed the OpenEXP format [3], as a response to the demand for higher color fidelity in the visual effect industry:

"16-bit integer-based formats typically represent color component values from 0 (black) to 1 (white), but do not account for over-range value (*e.g.*, a chrome highlight) that can be captured by film negative or other HDR displays. Conversely, 32-bit floating-point TIFF is often overkill for visual effects work. 32-bit FP TIFF provides more than sufficient precision and dynamic range for VFX images, but it comes at the cost of storage, both on disk and memory."

Similar arguments are used to justify the *half* format in Cg language. *Half* format is used to reduce storage cost, while computations are done with 32-bit FP formats, either by the CPU or the GPU.

In the remainder of this paper, this 16-bit FP format will be called F_{16} and the IEEE-754 32-bit single-precision FP format will be called F_{32}. Both formats are represented in Figure 2. F_{16} has a 1-bit sign, a 5-bit exponent with a bias of 15 and a 10-bit mantissa (with msb hidden), as in [9]. A number is interpreted exactly as in the other IEEE FP formats. The range of the format extends from $2^{-15} = 6 \times 10^{-5}$ to $2^{16} - 2^5 = 65504$. Denormal numbers are not used for the application considered here.

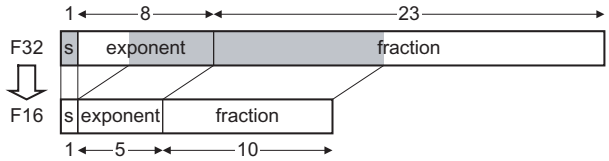


Figure 2. F_{32} and F_{16} floating-point numbers

From an architectural point of view, as many algorithms belong to the class of memory bounded problems, reducing the size of floating-point numbers could have a great impact on performance (just because twice less bytes have to be transfered); for algorithms with a small computational ratio (that is the number of mathematic computations per point divided by the number of memory accesses per point), their speed is limited by the memory bandwidth of the processor.

From an algorithmic point of view, the format can be tuned to an application, by allocating more bits to the mantissa to increase the precision, or to the exponent to increase the dynamic range. The bias can also be customized to favor numbers smaller or bigger than one. In embedded systems, in order to balance the hardware cost and the accuracy of algorithms, lighter FP coding with 15, 14 or even only 13 bits may be considered. Moreover, as F_{16} instructions require less logic than F_{32}, their implementation may lead to faster clock frequencies on an FPGA than F_{32}. Finally with such an approach, custom floating-point formats may be finely tuned to the application and to the embedded constraints.

Moreover, as interval arithmetic operations require from twice up to four times more computations than floating-point operations, where the speed of arithmetic operators depends on the width of their operands, reducing the number of bits, from 32 down to 16 could be a good trade-off between execution time and accuracy.

3.2 F16 intervals

When running interval algorithms on a RISC processor, even by using a library like PROFIL/BIAS [15, 16] to target real-time implementations (or at least quick implementation), one of the main problems lies in swapping the rounding mode used for floating-point computations. IEEE 754 defines four modes, and interval computations use two of them: towards $+\infty$ and towards $-\infty$. On most processors, a change of the rounding mode flushes its pipeline. The deeper the latter is, the bigger the resulting cycle penalty is.

FPGA and soft core processors can be an issue by

designing two FPUs; the first one is dedicated to computations with rounding towards $-\infty$, the second one towards $+\infty$. With two FPUs, changing the rounding mode is no more required. Another important point is that, on a 32-bit architecture, 16-bit numbers and 16-bit operators can be viewed as a natural way of doing parallel computations: two F_{16} numbers are stored inside a single 32-bit word and may be naturally used to define an F_{16} interval number. Such numbers are sent to an interval Floating-Point Unit (*iFPU*) where computations are done in parallel on the two bounds.

The soft core used for this implementation is the NIOS-II from Altera. In this customizable 32-bit RISC processor, new instructions, new data formats, and functions can be easily added in the C language.

Two Altera kits have been used: Cyclone and Stratix II. The Cyclone device has up to 20k Logic Elements and 288 kb SRAM. The Stratix II device has up to 60k logic elements, 2 Mb SRAM, and 36 DSP blocks which can implement 144 18 b×18 b multipliers. In our benchmarks, these DSP blocks introduce a significant difference in the multiplier performance. Both devices can use the NIOS-II soft-core CPU. The processor main features are summarized in Table 1. New instructions can be customized and added to the CPU ISA [4], as shown in Figure 3, in the *custom logic* block. The hardware operators for the customized instructions are described with VHDL or Verilog. Two types of operators can be defined. The combinational operators are used when the propagation delay is shorter than the processor cycle time. In that case, the defined interfaces are the 32-bit input data (dataa and datab) and the output data (result). When the propagation delay is greater than the clock cycle time, multi-cycle operations must be used. They have the same data interface than the combinational operators, plus clock and control signals: `clk` (processor clock), `clk_enable`, a global reset (`reset`), a `start` signal (active when the input data are valid) and a `done` signal (active when the result is available for the processor). Since the processor uses a 32-bit data interface, it is natural to define all our instructions as interval instructions: each one operates simultaneously on two 16-bit FP operands. This is a big advantage of using F_{16} operands as it doubles the throughput of operations. Using the customized instructions in a C program is straightforward. Two types of "`define`" are used as the instructions can have one or two input operands:

- `#define INST1(A) __builtin_custom_ini (Opcode_INSTR1, (A))`
- `#define INST2 (A, B) __builtin_custom_inii (Opcode_INSTR2, (A), (B))`

For both kits, the minimal clock frequency is 50 MHz. The maximal clock frequency depends on the complexity of the customized arithmetic instructions, and is higher for the Stratix II kit. The Quartus II software has been used to generate the FPGA configuration files from the NIOS-II file and the VHDL code for the customized instructions. All the benchmarks have been compiled with the Altera Integrated Development Environment (IDE), which uses the GCC tool chain. The optimizing option `-O2` has been used in release mode. Execution times have been measured with the `high_res_timer` that provides the number of processor clock cycles for the execution time. We only provide results for the Stratix II kit which is faster than the Cyclone kit.

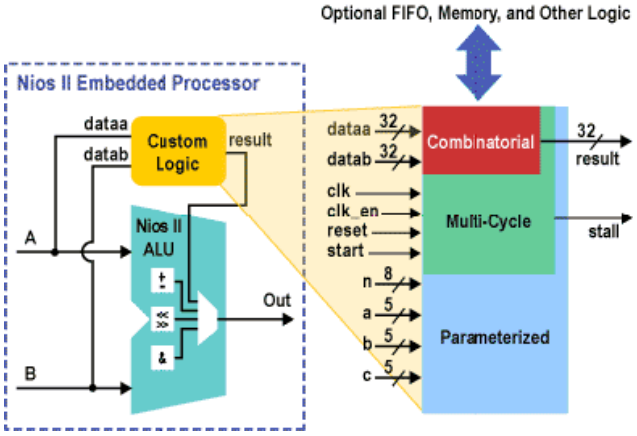


Figure 3. NIOS II architecture

Table 1 presents the fixed and tunable features of the NIOS-II processor. The clock frequency depends on the operator complexity and is usually in the range [100 MHz, 200 MHz]. Note that cache size are customizable from 2 and 4 kB up to 64 kB. Cache sizes have been set to the minimum to save space on the FPGA.

fixed features
32-bit RISC processor
branch prediction
dynamic branch predictor
barrel shifter
parameterized features
HW integer multiplication and division
4 KB instruction cache
2 KB data cache
customized instruction

Table 1. NIOS-II main features

4 Application

A network of $L = 5000$ sensors randomly distributed over a field of 100 m×100 m is considered, as well as a single source, such that $\boldsymbol{\theta}^* = (50\text{ m}, 50\text{ m})$, $A = 100$. The measurement noise is such that $e = 4$ dBm. Table 2 provides typical measurements with their associated interval. Only the sensors receiving significant amount of power ($y_\ell > 5$ (W)) participate to the localization.

sensor ℓ	measurement $[y_\ell]$ (W)
68	[9.303, 58.698]
741	[17.856, 112.664]
954	[18.644, 117.640]

Table 2. Example of measurements provided by three sensors of the network

4.1 Qualitative results: F_{32} vs F_{16} accuracy

In order to evaluate the impact of using F_{16} intervals in place of F_{32} intervals, the estimation algorithm has first been simulated with a Pentium M running the distributed constraint propagation algorithms using the Profil/Bias library with F_{32} arithmetic operators and functions (*log*, *exp*, *power*, and *sqrt*) and a F_{16} version of the same library (the size of the mantissa is reduced to 10 bits). When the algorithm converges (which requires between 2 and 3 cycles through the sensor network), a solution box is provided, which is guaranteed to contain the actual position of the source, provided that the hypotheses on the measurement noise were not violated.

Figure 4 represents two histograms (for F_{32} and F_{16}) of the absolute value of the localization error. Figure 5 provides two histograms (for F_{32} and F_{16}) of the maximum width of projection of the solution box on the $\boldsymbol{\theta}$-plane. For the simulation, 2000 runs with 5000 randomly distributed sensors were considered. The location of each sensor is represented using 16-bit floating-point values, in order to have exactly the same input data. The difference between the results obtained with F_{32} and F_{16} is thus only due to the smaller accuracy of F_{16} numbers (10 bits for F_{16} mantissa instead of 23 bits for that of the F_{32}).

The results obtained with F_{16} are very similar to those obtained with F_{32}. The average error and average width of the box containing $\mathbb{P}$ differ by less than 1%. For this kind of application, F_{16} provides a satisfying accuracy and dynamic range. The next section gives quantitative results on power consumption and on the required number of logical units on the FPGA.

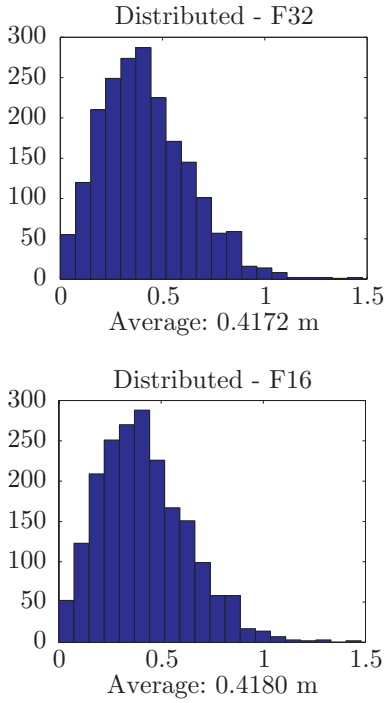


Figure 4. Localisation error (in meters): F_{32} vs F_{16}

4.2 Quantitative results: F_{16} interval vs F_{32} standard instructions

Table 3 provides a comparison between F_{32} and F_{16} for the number of blocks required for each instruction. The hardware description language (HDL) descriptions of the `DIV`, `POW`, `SQRT`, and `LOG` operators have been taken from [6]. For these operators the rounding mode *to nearest* has been replaced by the two rounding modes towards -∞ and towards $+\infty$.

Note that the 32-bit FPU only implements `ADD`, `MUL`, and `DIV`. Complex instructions like `POW`, `SQRT`, and `LOG` should be added in both cases. The implementation of `POW` is based on iterative calls to `MUL`. For these two instructions, the number of blocks is small, as DSP blocks incorporated on Stratix II FPGAs are used. `POW` and `MUL` respectively use two and four 18×18 DSP blocks. As can be seen in Table 3, an interval F_{16} processor is smaller than a scalar F_{32} processor.

Tables 4 and 5 present a comparison between a Pentium4 and a Pentium4-M running the F_{32} Profil/Bias library and a NIOS-II with F_{16} iFPU. For the

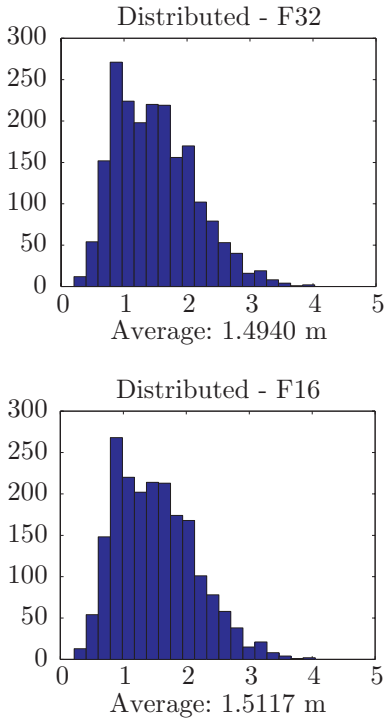


Figure 5. Diameter of the solution box (in meters): F_{32} vs F_{16}

NIOS-II, a cycle accurate chronometer is directly available. For the Pentium, two timers are available. The first one is based on the *Query Performance Counter* API which accuracy is better than 1 ms. The second one is a high performance counter (a 64-bit hardware register) that holds the total number of clock cycles elapsed from the boot.

As clock frequencies are very different, we prefer to use a normalized metric like the *cpi*, which is the number of processor cycles per iteration (total number of clock cycles divided by the number of iterations). It is a good metric to estimate the adequation between the algorithm and the architecture.

For embedded systems, the notion of performance does not only imply speed, but also power/energy consumption. Pentium 4 processors are fast but also very power consuming. If we look at the energy (another important metric for embedded systems) we can see that a NIOS2 processor on a StratixII device is twice more efficient. However, since no rounding mode switching is required, the NIOS2 *cpi* is smaller than the one of a Pentium 4, despite the large number of hardware optimizations present in the latter.

Two localization algorithms have been implemented.

Block	F_{16}	F_{32}
Frequency	50 MHz	50 MHz
CPU	2018	1831
32-bit FPU	-	5137
`ADD`	1441	-
`MED`	770	-
`MUL`	811	-
`DIV`	958	-
`POW`	439	782
`SQRT`	431	754
`EXP`	832	1289
`LOG`	898	1898
total	8590	11691

Table 3. Instructions size in block

The first one (Algorithm 1), presented in this paper (see Section 2.2), estimates the location of the source and the path loss exponent. The second one (Algorithm 2), assumes that the path loss exponent is known, which requires much less computations. Comparing these algorithms gives an idea of the price to pay (in terms of cycles per iteration) to estimate this additional parameter. Besides, the first of these two methods leads to data-dependent computation time. In our simulations, a satisfying convergence level was reached after three iterations of the algorithm in the whole sensor field. Before convergence, one iteration takes about 45000 cycles (to be compared to the 5000 cycles needed by the second algorithm) and about 9000 cycles after convergence, when fewer additional information can be extracted from the measurements.

	Pentium 4	Pemtium 4 - M	NIOS-II
frequency	2.4 GHz	1.6 GHz	50 MHz
time (ms)	11.4	17.1	186
average *cpi*	54800	54800	23400
power (W)	70 W	27 W	1 W
energy (mJ)	798 mJ	462 mJ	186 mJ

Table 4. Algorithm 1, unknown path loss exponent: P4 and P4-M vs NIOS-II for 5 iterations

Tables 4 and 5 provide results for Pentium 4, Pentium 4-M, and NIOS-II running the two algorithms for 5 iterations and 100 sensors participating to the localization. Estimating the path loss exponent requires seven times more computing power than not estimating this quantity. Comparing the NIOS-II to the Pentium 4, it can be seen that even for Algorithm 2, involving

	Pentium 4	Pentium 4 - M	NIOS-II
frequency	2.4 GHz	1.6 GHz	50 MHz
time (ms)	0.833	1.250	26
average *cpi*	4000	4000	2600
power (W)	70 W	27 W	1 W
energy (mJ)	58 mJ	34 mJ	26 mJ

Table 5. Algorithm 2, known path-loss exponent: P4 and P4-M vs NIOS-II for 5 iterations

mathematical functions (finely tuned by Intel inside its processors), NIOS-II is 4.3 times more energy efficient than Pentium 4 and 2.5 times more efficient than Pentium 4-M.

5 Conclusion

This paper has presented the evaluation of 16-bit floating-point operators and customizable floating-point formats for embedded systems performing source localization with interval computions.

While the implementation of such type of algorithms has some drawbacks on general-purpose processors (only one FPU with pipeline flushes because of rounding mode switching), the presented customized NIOS-II processor with two 16-bit FPUs, one for each rounding mode, tackles these drawbacks. As formats are customizable, interval FPUs can be tuned to the application (precision and dynamic range) with an energy-efficient implementation. Compared to a classical 32-bit implementation on a RISC computer, the proposed solution is 4.3 times more efficient.

Future works tend to develop high-level tools to perform an automatic design space exploration of the configurations to efficiently implement F_{16} format for interval computation into an FPGA. Right now, customization has been done at the instruction level, by adding new operations adapted to interval computations. A next step is to envision function customization that is to design a hardware accelerator implementing a full iteration of the localization algorithm. Some tools already exist like the Altera C2H compiler that directly *compiles* in hardware a C function. The current version of C2H is only able to compile integer functions, not FP ones. When such a kind of tool will be able to do so, processor customization will be available to non VHDL specialists.

References

[1] Location is everything: Positioning in wireless networks (a special issue). In A. Dogandzic, J. Riba, G. Seco, and A. Lee Swindlehurst, editors, *IEEE Signal Processing Magazine*, volume 22, 2005.

[2] Special issue on distributed constraint satisfaction. In B. Faltings and M. Yokoo, editors, *Artificial Intelligence*, volume 161, pages 1–250, 2005.

[3] *Technical Introduction to OpenEXR*, 2006. http://www.openexr.com/TechnicalIntroduction.pdf.

[4] Altera. *NIOS Custom Instructions, Tutorial*, 2002. http://www.altera.com/literature/tt/tt_nios_ci.pdf.

[5] R. Bejar, C. Fernandez, M. Valls, C. Domshlak, C. Gomes, B. Selman, and B. Krishnamachari. Sensor networks and distributed CSP: Communication, computation and complexity. *Artificial Intelligence Journal*, 161(1-2):117–148, 2005.

[6] J. Detrey and F. de Dinechin. *FPLibrary: A VHDL Library of Parametrisable Floating-Point and LNS Operators for FPGA*. ENS Lyon, 2006. http://perso.ens-lyon.fr/jeremie.detrey/FPLibrary/.

[7] D. Etiemble, S.Bouaziz, and L. Lacassagne. Customizing 16-bit floating point instructions on a NIOS II processor for FPGA image and media processing. In *Proc. Estimedia*, New York, 2005.

[8] A. O. Hero III and D. Blatt. Sensor network source localization via projection onto convex sets (POCS). In *Proceedings of ICASSP*, 2005.

[9] IEEE. *DRAFT Standard for Floating-Point Arithmetic P754*, 2006. Draft 1.2.5, http://754r.ucbtest.org/drafts/754r.pdf.

[10] Intel. Desktop performance and optimization for intel pentium 4 processor. Technical Report 249438-01, 2001.

[11] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control*, 74(18):1772–1782, 2001.

[12] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer-Verlag, London, 2001.

[13] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

[14] M. Kieffer and E. Walter. Centralized and distributed source localization by a network of sensors using guaranteed set estimation. In *Proceedings of ICASSP*, volume 4, pages 977–980, 2006.

[15] O. Knüppel. PROFIL - programmer's runtime optimized fast interval library. Technical Report 93.4, Institut für Informatik III, Technische Universität Hamburg-Harburg, Germany, 1993. Available at: ftp://ftp.ti3.tu-harburg.de/pub/reports/report93.3.ps.Z.

[16] O. Knüppel. PROFIL/BIAS – A fast interval library. *Computing*, 53:277–287, 1994.

[17] R. Kolla, A. Vodopivec, and J. Wolff Von Gudenberg. The IAX architecture: Interval arithmetic extension.

Technical report, 1999. http://wwwi2.informatik.uni-wuerzburg.de/mitarbeiter/wvg/Public/iax.ps.gz.

[18] U. Kulisch and W. L. Miranker. The arithmetic of digital computer: A new approach. *Siam Review*, 28(1), 1986.

[19] L. Lacassagne, D. Etiemble, and S. Ould Kablia. 16-bit floating point instructions for embedded multimedia applications. In *Proc. IEEE Computer Architecture and Machine Perception*, Palermo, 2005.

[20] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. Cg: A system for programming graphics hardware in a c-like language. In *Proceedings of SIGGRAPH 2003*, 2003.

[21] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[22] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 1990.

[23] Y. Okumura, E. Ohmori, T. Kawano, and K. Fukuda. Field strength ans its variability in VHF and UHF land-mobile radio service. *Rev. Elec. Commun. Lab.*, 16:9–10, 1968.

[24] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal. Locating the nodes. *IEEE Signal Processing Magazine*, 22(4):54–69, 2005.

[25] M. G. Rabbat and R. D. Nowak. Decentralized source localization and tracking. In *Proc. ICASSP*, 2004.

[26] A. H. Sayed, A. Tarighat, and N. Khajehnouri. Network-based wireless location. *IEEE Signal Processing Magazine*, 22(4):24–40, 2005.

[27] G. Sun, J. Chen, W. Guo, and K. J. Ray Liu. Signal processing techniques in network-aided positioning. *IEEE Signal Processing Magazine*, 22(4):12–23, 2005.

[28] J. Wolff von Gudenberg. Hardware support for interval computation. In G. Alefeld, A. Frommer, and B. Lang, editors, *Scientific Computing and Validated Numerics*, pages 32–37. Akademie-Verlag, Berlin, Germany, 1996.

Guaranteed Robust Tracking with Flatness Based Controllers Applying Interval Methods

Marco Kletting, Eberhard P. Hofer
Institute of Measurement, Control, and Microtechnology
University of Ulm
D-89081 Ulm, Germany
{Marco.Kletting, Ep.Hofer}@uni-ulm.de

Felix Antritter
Institute of Automatic Control,
University of Erlangen-Nürnberg
Cauerstrasse 7
D-91058 Erlangen, Germany
Felix.Antritter@rt.eei.uni-erlangen.de

Abstract

Flatness based tracking controller design (see e.g. [4, 5, 16]) is one of the most important tools for the control of nonlinear systems. A drawback of this approach is the lack of methods for the robustness analysis of such controllers with respect to uncertain parameters in the plant. In [1] the application of interval methods has been proposed for the guaranteed robustness analysis of flatness based tracking controllers. This approach allows to explicitly calculate the deviations from the reference trajectory which are caused by uncertain parameters in the plant in a guaranteed way. In this contribution the analysis using interval methods is extended to the case when a nonlinear tracking observer is necessary to estimate unmeasured states. Furthermore it is shown that unknown sensor offsets can be included into this robustness framework. The approach is illustrated for a magnetic levitation system.

1 INTRODUCTION

Flatness based controller design [4, 5, 16] is a powerful tool for motion planning and trajectory tracking for linear and nonlinear systems. Especially, for nonlinear systems there is a wide acceptance of this approach, which has been applied successfully to numerous problems of industrial relevance. However, a major drawback is the lack of techniques that allow to investigate the robustness of flatness based tracking controllers against, e.g., parameter uncertainties in the plant and measurement uncertainties due to non-ideal sensors.

Roughly speaking, the flatness property of a nonlinear system is characterized by the existence of an — possibly fictitious — flat output that allows a differential parameterization of the states and inputs. Based on the differential parameterization a tracking controller for a given reference trajectory for the flat output can be designed. In general, not all system states which are necessary to implement the tracking controller can be measured . In this case a nonlinear tracking observer as proposed in [6] can be used. These relations are discussed in a general manner for single-input systems and are applied to a magnetic levitation system, where only the load position can be measured.

Interval methods [4,5] are used to analyze the dynamic behavior of the controlled system, which is described by a system of nonlinear differential equations.

Applying these techniques the maximum admissible range of parameter uncertainties in the plant is determined such that the position of the load along the prescribed trajectory is guaranteed to be within specified tolerances.

More detailed, subintervals of the parameter uncertainties are considered for a verified integration [1] over the desired time span. A subinterval is admissible if the resulting enclosures over the complete time span lie completely inside the specified tolerances for robustness. If the enclosures are completely outside the specified tolerances for at least one point of time, the corresponding subinterval is not admissible. Further splitting is required to decide about the admissibility of all remaining intervals. In addition to un-

certainties of parameters of the plant also interval uncertainties of the initial conditions and the available measured data can be considered. For the verified integration a Taylor model based solver as implemented in COSY-VI [3] is used.

The methodology for robustness analysis is restricted to single-input systems in this contribution. However, with obvious extensions it can also be applied to multi-input systems and a wide class of controllers and dynamical systems.

This paper is organized as follows. In Section 2.1, flatness based control and the construction of tracking observers is introduced shortly. In Section 3 these relations are applied to a magnetic levitation system. Section 4 describes briefly the Taylor model based verified integration of nonlinear uncertain systems, which is required for the Robustness analysis presented in Section 5. Simulation results are shown in Section 6. Finally, conclusions and an outlook on future research are given in Section 7.

2 FLATNESS BASED CONTROLLER DESIGN

2.1 Flatness

Flatness based controller design has been introduced e.g. in [4] (differential algebraic setting) and [5] (differential geometric setting). Various aspects of flatness are illustrated e.g. in [16]. In this contribution the following relations for nonlinear single input systems are used, where explicitely the dependence of the relations on the parameters are stated: For a flat system

$$\dot{x} = f(p, x, u) \tag{1}$$

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}$ and the parameter vector $p \in \mathbb{R}^{n_p}$ the flatness property implies the existence of a flat output $y_f \in \mathbb{R}$, such that

$$y_f = h_f(p, x) \tag{2}$$

$$x = \psi_x(p, y_f, \dot{y}_f, \ldots, y_f^{(n-1)}) \tag{3}$$

$$u = \psi_u(p, y_f, \dot{y}_f, \ldots, y_f^{(n)}) \tag{4}$$

holds, with h_f, ψ_u, ψ_x smooth at least on an open subset of $\mathbb{R}$, $\mathbb{R}^n$ and $\mathbb{R}$ respectively. Introducing the new coordinates

$$\zeta = (\zeta_1, \ldots, \zeta_n) = (y_f, \dot{y}_f, \ldots, y_f^{(n-1)}) \,, \tag{5}$$

the flat system (1) can be transformed via the well defined diffeomorphism

$$\zeta = \Phi(p, x) \tag{6}$$

into controller normal form

$$\begin{aligned} \dot{\zeta}_i &= \zeta_{i+1}, \qquad i = 1, 2, \ldots n-1 \\ \dot{\zeta}_n &= \alpha(p, \zeta, u) \,. \end{aligned} \tag{7}$$

Setting $v = y_f^{(n)}$ yields

$$u = \psi_u(\zeta, v) \tag{8}$$

in view of (4) and (5). In [8] it has been shown that

$$\alpha(p, \zeta, \psi_u(p, \zeta, v)) = v \tag{9}$$

holds and thus by application of the feedback law (8), system (1) is diffeomorphic to the Brunovský normal form

$$\begin{aligned} \dot{\zeta}_i &= \zeta_{i+1}, \qquad i = 1, 2, \ldots n-1 \\ \dot{\zeta}_n &= v \end{aligned} \tag{10}$$

with new input v.

2.2 Flatness Based Feedforward Controller

Due to the derived relations in Section 2.1 a (sufficiently smooth) reference trajectory $y_{f,d} : [t_0, t_0 + T] \to \mathbb{R}$ for the flat output y_f can be assigned almost arbitrarily (excluding singularities of the differential parameterization (3)–(4)). If the reference trajectory $y_{f,d}$ satisfies the boundary conditions

$$x(t_0) = \psi_x(p_0, y_{f,d}(t_0), \dot{y}_{f,d}(t_0), \ldots, y_{f,d}^{(n-1)}(t_0)) \tag{11}$$

then a corresponding feedforward controller that provides $y_f(t) = y_{f,d}(t)$ for $t \in [t_0, t_0 + T]$ is given by

$$u_d(t) = \psi_u(p_0, y_{f,d}(t), \dot{y}_{f,d}(t), \ldots, y_{f,d}^{(n)}(t)) \;. \tag{12}$$

For (11) and (12) it has been assumed that the parameters of the plant (1) match a nominal parameter vector p_0.

2.3 Flatness Based Tracking Controller design

To stabilize the tracking of a given reference trajectory $y_{f,d}$ for the flat output, the tracking error e is introduced as

$$e = y_f - y_{f,d} = \zeta_1 - \zeta_{1,d} \tag{13}$$

In view of (10) it follows that

$$e^{(i)} = \zeta_{i+1} - \zeta_{i+1,d}, \quad i = 0, 1, \ldots, n-1 \;. \tag{14}$$

Thus, when setting the new input v in (10) to

$$v = \dot{\zeta}_{n,d} - \sum_{i=0}^{n-1} \lambda_i (\zeta_{i+1} - \zeta_{i+1,d}) = \dot{\zeta}_{n,d} - \sum_{i=0}^{n-1} \lambda_i e^{(i)} \;, \tag{15}$$

the tracking error obeys the differential equation

$$0 = e^{(n)} + \sum_{i=0}^{n-1} \lambda_i e^{(i)} \tag{16}$$

which can be achieved to be stable by suitable choice of the λ_i. Substituting (15) into the differential parameterization (4) of the input yields in view of (5) the feedback law

$$u = \psi_u(p, y_f, \dot{y}_f, \ldots, y_f^{(n-1)}, y_{f,d}, \dot{y}_{f,d}, \ldots, y_{f,d}^{(n)}) \ . \quad (17)$$

Using the diffeomorphism (6), the feedbacklaw (17) can be implemented as

$$u = \psi'_u(p_0, x, y_{f,d}, \dot{y}_{f,d}, \ldots, y_{f,d}^{(n)}) = \psi''_u(p_0, x, t) \ , \quad (18)$$

where again the plant parameters p are assumed to be equal to the nominal parameter vector p_0. As a consequence, for the feedback controller (18), the controlled system can be summarized as

$$\dot{x} = f(p, x, \psi''_u(p_0, x, t)) = f_{fb}(p, x, t) \ , \quad (19)$$

where $p \neq p_0$ can occur due to not exactly known parameters. To improve the robustness of the tracking controller an integral error feedback is often introduced, i.e. the error feedback (15) is extended according to

$$\begin{aligned} \dot{e}_I &= \zeta_1 - \zeta_{1,d} \quad (20) \\ v &= \dot{\zeta}_{n,d} - \sum_{i=0}^{n-1} \lambda_i e^{(i)} - \lambda_{-1} e_I \ . \end{aligned}$$

This feedback can clearly be implemented as a state feedback of the kind

$$\begin{aligned} \dot{e}_I &= h_f(p, x) - y_{f,d}(t) \quad (21) \\ u &= \psi''_{u,I}(p_0, x, e_I, t) \ . \end{aligned}$$

2.4 Tracking using a Nonlinear Tracking Observer

For the implementation of the feedback (18), in general, all states have to be available for measurement. If only the output

$$y = h(p, x) \quad (22)$$

with $y \in \mathbb{R}^m$ is available for measurement, a nonlinear tracking observer with time varying observer gain $L(t)$

$$\begin{aligned} \dot{\hat{x}} &= f(p_0, \hat{x}, u) + L(t)(y - h(p_0, \hat{x})) \quad (23) \\ &= f(p_0, \hat{x}, u) + L(t)(h(p, x) - h(p_0, \hat{x})) \\ &= f_{obs}(p, x, \hat{x}, u, t) \end{aligned}$$

as proposed in [6] can be used. The observer (23) basically consists of a model of the plant and a feedback of the difference of the measured output and the estimated output. For the model of the plant also the nominal parameter values p_0 are used. The time varying observer gain $L(t)$ is designed such that the linearization of the estimation error dynamics about the reference trajectory $y_{f,d}$ which result to

$$\Delta\dot{\hat{x}} - \Delta\dot{x} = (A(t) - LC(t))(\Delta\hat{x} - \Delta x) \quad (24)$$

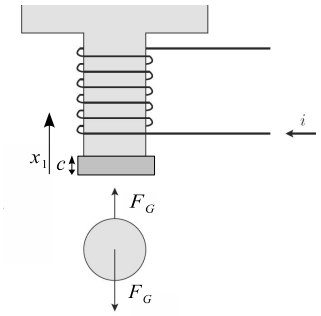


Figure 1. Magnetic levitation system.

with

$$A(t) = \frac{\partial f}{\partial x}\Big|_{x_d, u_d}, \qquad C(t) = \frac{\partial h}{\partial x}\Big|_{x_d, u_d} \quad (25)$$

are stable. For the stabilization of (24), i.e. of the estimation error dynamics in the vicinity of the reference trajectory $y_{f,d}$, methods for linear time varying systems as proposed in [7] can be used. Using the tracking observer (23) the feedback (18) can be estimated using the observer states $\hat{x}$

$$\hat{u} = \hat{\psi}(p_0, \hat{x}, t) \ . \quad (26)$$

Thus, the controlled system can be summarized as

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} f(p, x, \hat{\psi}(p_0, \hat{x}, t), t) \\ f_{obs}(p_0, x, \hat{x}, \hat{\psi}(p_0, \hat{x}, t), t) \end{bmatrix} = f_{fbo}(p, x, \hat{x}, t) \ . \quad (27)$$

It can easily be deduced that when using an observer together with the controller (21) which includes integral error feedback the following structure results

$$\begin{aligned} \begin{bmatrix} \dot{e}_I \\ \dot{x} \\ \dot{\hat{x}} \end{bmatrix} &= \begin{bmatrix} h_f(p, x) - y_{f,d}(t) \\ f(p, x, \hat{\psi}(p_0, \hat{x}, t), t) \\ f_{obs}(p_0, x, \hat{x}, \hat{\psi}(p_0, \hat{x}, t), t) \end{bmatrix} \\ &= f_{fbo,I}(p, x, \hat{x}, e_I, t) \ . \quad (28) \end{aligned}$$

The controlled systems (27) and (28) have a similar structure as (19). This structure can be analysed using the methods discussed in Sections 4 and 5.

Some additional modifications for the tracking controllers with observer can be introduced that do not change the resulting structure and will be discussed in Section 3.

3 MAGNETIC LEVITATION SYSTEM

A simplified model of a magnetic levitation system (see Figure 1) is given by [13]

$$\begin{aligned} \dot{x}_1 &= x_2 \quad (29) \\ \dot{x}_2 &= \frac{k}{m} \frac{u^2}{(c - x_1)^2} - g \ . \end{aligned}$$

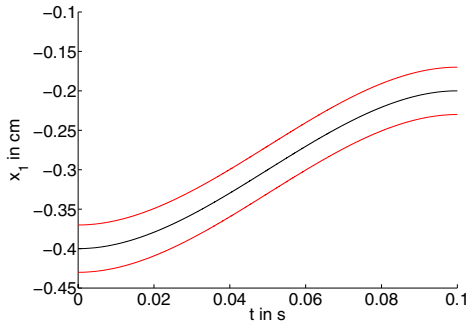


Figure 2. Reference trajectory for the position x_1 (black) and the bounds for admissible deviations (grey).

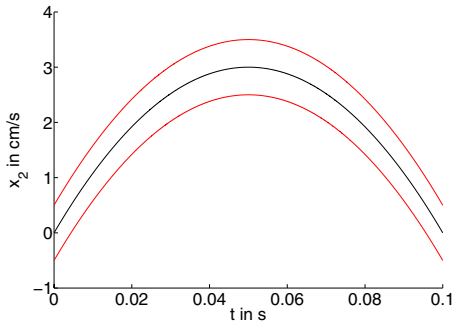


Figure 3. Reference trajectory for the velocity x_2 (black) and the bounds for admissible deviations (grey).

The system is already given in controller normal form and thus a flat output of (29) is given by

$$y_f = x_1 \ . \tag{30}$$

The relations (3)–(4) can directly be derived from (29)

$$(x_1, x_2) = (y_f, \dot{y}_f) \tag{31}$$

$$u = (c - y_f)\sqrt{\frac{m}{k}(\ddot{y}_f + g)} \ . \tag{32}$$

For the load of the levitation system a set point change is considered, i.e. a tajectory has to be planned such that the following boundary conditions are satisfied

$$\begin{aligned}(x_1(0\,\mathrm{s}), x_2(0\,\mathrm{s})) &= (-0.4\,\mathrm{cm}, 0\,\frac{\mathrm{cm}}{\mathrm{s}}) \\ (x_1(0.1\,\mathrm{s}), x_2(0.1\,\mathrm{s})) &= (-0.2\,\mathrm{cm}, 0\,\frac{\mathrm{cm}}{\mathrm{s}}) \ .\end{aligned} \tag{33}$$

In view of the differential parameterization (31) this yields the following boundary conditions for a corresponding trajectory $y_{f,d}$ for the flat output

$$\begin{aligned} y_{f,d}(0\,\mathrm{s}) &= -0.4\,\mathrm{cm}, & \dot{y}_{f,d}(0\,\mathrm{s}) &= 0\,\frac{\mathrm{cm}}{\mathrm{s}} \\ y_{f,d}(0.1\,\mathrm{s}) &= -0.2\,\mathrm{cm}, & \dot{y}_{f,d}(0.1\,\mathrm{s}) &= 0\,\frac{\mathrm{cm}}{\mathrm{s}} \end{aligned} \tag{34}$$

which can be satisfied by assigning for $y_{f,d}$ a third order polynomial. The resulting trajectory for y_f can be seen in Figure 2.

Based on the results in Section 2.1 a tracking controller for system (29) is given by

$$u = (c - x_1) \cdot \sqrt{\frac{m}{k}(\ddot{y}_{f,d} - \lambda_1(x_2 - y_{f,d}) - \lambda_0(x_1 - y_{f,d}) + g)} \ . \tag{35}$$

It is assumed that the flat output of (29) is available for measurement, i.e.

$$y = h(x) = x_1 + \epsilon \ , \tag{36}$$

where, however, ϵ describes an unknown but constant sensor offset. For the given output a nonlinear tracking observer, as discussed in Section 2.4, can be derived. It has the form

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 - l_1(t)(x_1 + \epsilon - \hat{x}_1) \\ \dot{\hat{x}}_2 &= \frac{k}{m}\frac{u^2}{(c - \hat{x}_1)^2} - g - l_2(t)(x_1 + \epsilon - \hat{x}_1) \ , \end{aligned} \tag{37}$$

where the unknown sensor offset ϵ in (36) is explicitely included in the model. The feedback law (35) which stabilizes the tracking can then be estimated as

$$\hat{u} = (c - x_1 + \epsilon) \cdot \sqrt{\frac{m}{k}(\ddot{y}_{f,d} - \lambda_1(\hat{x}_2 - y_{f,d}) - \lambda_0(x_1 + \epsilon - y_{f,d}) + g)} \ , \tag{38}$$

where the measured output (36) was used to estimate x_1 and x_2 is estimated using the observer state $\hat{x}_2$. A tracking controller with integral error feedback as discussed in Section 2.1 is given by (using again the observer (37) and the measured output (36))

$$\begin{aligned} \dot{e}_I &= x_1 + \epsilon - y_{f,d} \\ \hat{u} &= (c - x_1 + \epsilon) \cdot \\ &\quad \Big(\frac{m}{k}(\ddot{y}_{f,d} - \lambda_1(\hat{x}_2 - y_{f,d}) - \lambda_0(x_1 + \epsilon - y_{f,d}) \ldots \\ &\quad \ldots - \lambda_{-1} e_I) + g)\Big)^{-\frac{1}{2}} \end{aligned} \tag{39}$$

in view of (21). With the parameter vector $p = (k, m, c, g, \epsilon)$ the controlled systems (29), (37)–(38) and (29), (37), (39) exhibit the structure as in (27) and (28) respectively.

Assume that there are constraints for the at most tolerable deviations from the reference trajectory for the controlled system which are specified in the following manner

$$|x_i(t) - x_{i,d}(t)| < \delta_i, \quad i = 1, 2; \ \forall\, t \in [0\,\mathrm{s}, 0.1\,\mathrm{s}] \ . \tag{40}$$

In the sequel it will be shown that, using interval methods, it is possible to determine the admissible parameter interval $[\underline{p}, \overline{p}]$ with $p_0 \in [\underline{p}, \overline{p}]$ such that the tracking controllers can meet the specification (40). This question can be solved using the tools introduced in the next section. Also a more formal statement of the robustness requirement will be given.

4 VERIFIED INTEGRATION BASED ON TAYLOR MODELS

The controlled systems (27) and (28) respectively can be described by a set of time varying nonlinear ordinary differential equations

$$\dot{x}(t) = f_x(x(t), p(t), t), \tag{41}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector (including eventually the controller state for the integral error feedback) and $p \in \mathbb{R}^{n_p}$ the parameter vector. The parameter vector p and the initial conditions $x(0)$ are assumed to be uncertain with $p \in [\underline{p}, \overline{p}]$ and $x(0) \in [\underline{x}(0), \overline{x}(0)]$. If the parameters may vary over time within their bounds and if upper and lower bounds of the variation rate are known then

$$\dot{p}(t) = \Delta p \quad \text{with} \quad \Delta p \in [\Delta\underline{p}, \Delta\overline{p}] \tag{42}$$

holds.
The state vector can be extended by the parameter vector according to

$$\begin{aligned} \dot{z}(t) =& f(z(t), u(t)) \text{ with } z(t) = [x(t)^T, p(t)^T]^T \text{ and} \\ & f = \begin{bmatrix} f_x(x(t), p(t)) \\ \Delta p \end{bmatrix} \end{aligned} \tag{43}$$

with $f : D \mapsto \mathbb{R}^n$, $D \subset \mathbb{R}^n = \mathbb{R}^{n_x} \times \mathbb{R}^{n_p}$. Uncertain parameters which are time-invariant are described by $\Delta p = 0$.

For the robustness analysis a verified integration of the system model has to be performed. In this paper a Taylor model based integrator as implemented in COSY VI is used. The goal is to find a Taylor model

$$T = P_{\rho,z} + I_z \tag{44}$$

consisting of Taylor polynomial P and interval bounds I for the remainder error for the flow of the differential equation

$$\dot{z}(t) = f(z(t), t) \tag{45}$$

in the time interval $[t_0; t_1]$ with $z(t_0) \in [z_{01}; z_{02}]$, where $z(t_0)$ may also be a Taylor model [3, 10, 11]. The initial interval box at $t = 0$ can be expressed as a Taylor model

$$z(0) = c_0 + Dz_0 \text{ with } z_{i,0} \in [-1; 1], i = 1, 2, \ldots n \tag{46}$$

where c is the midpoint of $z(0)$ and D is a diagonal Matrix with $d_{i,i} = 0.5 \cdot \mathrm{diam}(z(0))$. In contrast to algorithms such as implemented in VNODE [15] not only an Expansion in time t but also in the initial conditions z_0 is performed, which reduces the dependency problem and the wrapping effect. In each time-step the set is described by a Taylor model,which allows for a more flexible enclosure of non-convex sets.

First equation (45) is rewritten into Integral form

$$z(t) = z(t_0) + \int_{t_0}^{t} f(z(\tau), \tau) d\tau \ . \tag{47}$$

Which leads to a fix-point equation

$$A(z)(t) = z(t_0) + \int_{t_0}^{t} f(z(\tau), \tau) d\tau \ . \tag{48}$$

The goal is now to find a Taylor model of order ρ for the flow of the differential equation, such that

$$A(P_{\rho,z} + I_z) \subseteq P_{\rho,z} + I_z \tag{49}$$

is fulfilled. A more detailed description of the algorithm can be found in [3].

As already mentioned, the expansion in initial conditions reduces the dependency problem and the wrapping effect. In order to limit a long-term growth of the truncation error and for further reduction of overestimation the following strategies can be applied:

- Shrink Wrapping: The Interval remainder is absorbed in the polynomial part [11]
- Preconditioning: The solution of the ODE is studied in a different coordinate system in order to minimize long-term error growth [10]
- State-space extension by the remainder term during simulation
- Splitting of the remainder term during simulation [12]
- Splitting of the reference domain during simulation

5 ROBUSTNESS ANALYSIS OF THE TRACKING CONTROLLER

The goal is to determine parameter values

$$\begin{aligned} \Omega_{in} = & \\ & \left\{ \begin{bmatrix} x_0 \\ p_0 \end{bmatrix} = \mathbf{z(0)} \middle| \, |x_i(t) - x_{i,d}(t)| \leq \delta_i \forall t \in [t_0, t_0 + T], \right. \\ & \left. i = 1, 2 \right\}, \end{aligned} \tag{50}$$

for which it can be guaranteed that the conditions for robustness in equation (40) are fulfilled and those parameter values

$$\Omega_{out} = \left\{ \begin{bmatrix} x_0 \\ p_0 \end{bmatrix} = \mathbf{z}(\mathbf{0}) \middle| |x_i(t) - x_{i,d}(t)| > \delta_i \forall t \in [t_0, t_0 + T], \; i = 1, 2 \right\}, \tag{51}$$

for which it can be guaranteed that these conditions are not fulfilled. Uncertain parameters are given by p, $x(0)$ are the possibly uncertain initial conditions and δ_i are the allowed tolerances around the reference trajectories of the position x_1 and velocity x_2.

The determination of Ω_{in} and Ω_{out} can be done by splitting the vector $z(0)$ in subintervals

$$\tilde{z}_l(0), l = 1, 2, \ldots, L, \quad \bigcup_{l=1}^{L} \tilde{z}_l(0) = z(0) \tag{52}$$

and performing a verified simulation with a Taylor model based ode-solver as implemented in COSY VI. The algorithm is described in Fig. 4. First an interval $\tilde{z}_l(0)$ is selected for the robustness analysis, then a splitting criterion is evaluated and the selected interval box is split. For the split intervals a verified integration of the system model is performed. Then, three different cases have to be distinguished:

1. If for any $t \in [0, T]$, the resulting trajectory is completely outside the specified tolerances the corresponding interval box is inconsistent and can be deleted.

2. If on the other hand the resulting trajectory lies completely inside the tolerance for all $t \in [0, T]$, the corresponding interval box is admissible.

3. Intervals which lead to trajectories which are not completely outside but also not completely inside for all $t \in [0, T]$ have to be split further until a user given maximum number of splitting operations is reached.

In the following, criteria for selection and splitting of the intervals shall be discussed in more detail. Each subinterval $\tilde{z}_l(0)$ can again be expressed as a Taylor model with the unit box z_0 as reference domain according to

$$\tilde{z}_l(0) = \tilde{c}_{0l} + \tilde{D}_l z_0 \text{ with } z_{i,0} \in [-1; 1], i = 1, 2, \ldots n \quad l = 1, 2, \ldots, L \tag{53}$$

where $\tilde{c}_{0l}$ is the midpoint of $\tilde{z}_l(0)$ and $\tilde{D}_l$ is a diagonal Matrix with $\tilde{D}_l = 0.5 \cdot \mathrm{diag}\left(\mathrm{diam}(\tilde{z}_l(0))\right)$.

If $L > 1$, the most appropriate subinterval for the splitting has to be selected at first. This could be the interval with the largest pseudo-volume, which is the product of the diameters of the resulting interval vector components. Another strategy is to calculate the pseudo volume of the interval enclosure of the Taylor model $\tilde{T}_l(z_0)$ resulting from each subinterval $\tilde{z}_l(0)$ in the last integration step of the preceding integration and select the subinterval $\tilde{z}_l(0)$ which led to the largest pseudo volume.

After the selection of an interval $\tilde{z}_l(0)$ a splitting direction has to be determined by checking the sensitivity of the Taylor model $\tilde{T}_l(z_0)$ from the selected interval $\tilde{z}_l(0)$ at the last integration step of the previous integration with respect to each component $z_{i,0}$,$i = 1 \ldots n$ of the reference domain.

The component μ of the reference domain z_0 for which the Taylor model $\tilde{T}_l(z_0)$ is most sensitive is determined by the following heuristics. First all $w_{i,j}$

$$w_{i,j} = \mathrm{diam}(z_{i,0}) \left| \frac{\partial \tilde{T}_{lj}(z_0)}{\partial z_{i,0}} \Big|_{z_0 = \mathrm{mid}(z_0)} \right| \quad i, j = 1, \ldots, n,$$

have to be calculated and the component μ is determined by

$$\mu = \underbrace{\mathrm{argmax}}_{l=,\ldots,n} \sum_{i=1}^{n} w_{i,j} \ . \tag{54}$$

As the intervals $\tilde{z}_l(0)$ and z_0 are related by equation (53) the interval $\tilde{z}_l(0)$ selected for splitting is also split in the component μ. In equation (54), the results are the coefficients of the linear parts of the Taylor model $T(z_0)$ multiplied by the factor 2, because $\mathrm{mid}(z_{i,0}) = 0$ and $\mathrm{diam}(z_{i,0}) = 2$ for $z_{i,0} \in [-1; 1]$, $i = 1 \ldots, n$.

6 SIMULATION RESULTS

For the simulation the nominal parameters of system (29) have been assumed to be (see [9])

$$k_0 = 58.041 \frac{\mathrm{kg\,cm^3}}{\mathrm{s^2\,A^2}}, \qquad g_0 = 981 \, \frac{\mathrm{cm}}{\mathrm{s^2}} \tag{55}$$
$$m_0 = 0.0844 \, \mathrm{kg}, \qquad c_0 = 0.11 \, \mathrm{cm}$$

The algorithm described in the previous section has been applied to the magnetic levitation system which is controlled using the tracking controllers [(37), (38)] and [(37), (39)] respectively. For simplicity the parameters of the levitation system are assumed to match then nominal ones in (55). Only the parameter k is assumed to be uncertain but bounded by the interval $k \in [54.042; 62.042] \frac{\mathrm{kg\,cm^3}}{\mathrm{s^2\,A^2}}$. The sensor is assumed to have an unknown offset ϵ which is assumed to be bounded by $\epsilon \in [-0.01; 0.01]$ cm.The order of the Taylor models was chosen to be 6 in time and initial states. Additionally QR-Preconditioning was applied. The tolerances for the deviations from the reference trajectory (see (40)) are: $\delta_1 \in [-0.03; 0.03]$ cm, $\delta_2 \in [-0.5; 0.5] \, \frac{\mathrm{cm}}{\mathrm{s}}$. δ_1 is chosen very small as the final equililibrium position is desired to be approached very exactly.

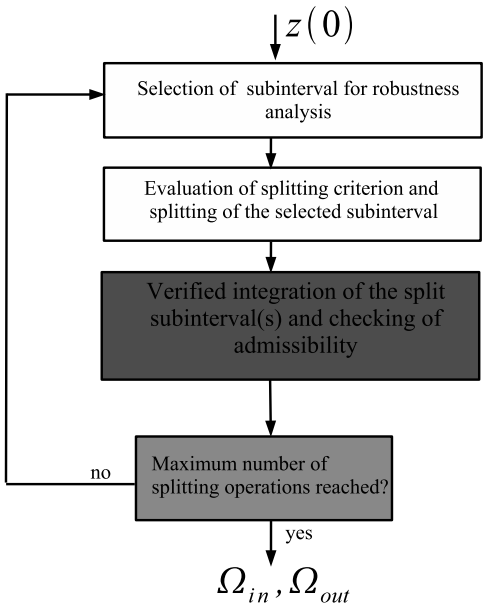


Figure 4. block diagram of the algorithm for the determination of the admissible parameters

To speed up the convergence of the estimation error the observer (37) has been initialized using the measured output (36), i.e.

$$\hat{x}_1(0) = y(0) \tag{56}$$

As x_2 cannot be measured the observer state $\hat{x}_2$ is initialized with the reference trajectory, i.e.

$$\hat{x}_2(0) = x_{2,d}(0) = \dot{y}_{f,d}(0) \tag{57}$$

The simulation results for the controller [(37), (38)] can be seen in Figure 5. For the consistent set of parameters the robustness specifications (40) are met. For the inconsistent parameters the specifications are not met. Remaining undecided parameter sets could be split up further, if necessary. It can be deduced that the controller [(37), (38)] cannot meet the specification (40) for the complete uncertainty intervals of k and ϵ and thus the parameters k and ϵ would have to be determined more exactly if the specifications have to be met in any case.

In Figure 6 the resulting consistent parameters are compared for the controllers [(37), (38)] and [(37), (39)]. It can be seen that for the desired specifications the two controllers yield different subsets of the uncertainty intervals. It can be concluded e.g. that if there is a large uncertainty in the offset ϵ of the sensor and the parameter k can be determined very well, then the controller [(37), (38)] should be preferred (indicated by the dashed box). If on the other hand the uncertainties in the sensor offset are small but the uncertainty in the parameter k is relatively large, then the controller with integral error feedback should be preferred. Thus the proposed robustness analysis can indeed yield hints on the choice of the used controller.

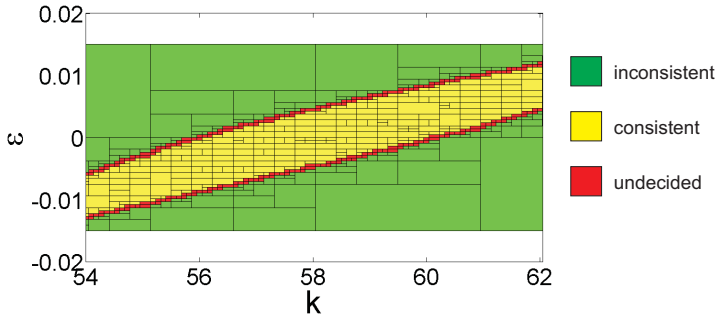


Figure 5. Simulation results for controller without integral error feedback.

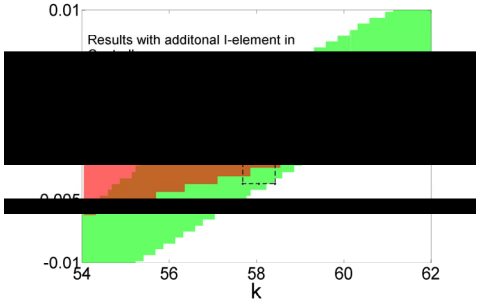


Figure 6. Comparison of the results.

It should be mentioned however, that the resulting parameter sets as shown in Figures 5 and 6 strongly depend on the specifications (40). This can be verified when looking at Figure 7 and 8. Here the resulting trajectories using the controller [(37), (39)] for $k = 56.5$ and $\epsilon = 0.004$ —a parameter combination which belongs to the set unadmissible parameter values— are depicted. It can be seen that the robustness specification is violated only by the velocity and mostly in the first part of the trajectory. As a consequence, if the tolerable deviation e.g. on the velocity in the first part of the trajectory can be relaxed or a specification which allows some kind of a transition time for the states — as also depicted in Figure 7 and 8 — a much larger admissible parameter set would result for this controller. With the proposed approach also such a specification could be investigated.

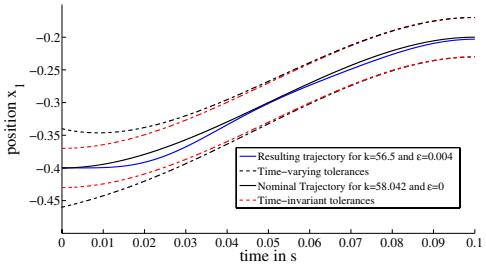


Figure 7. Modified tolerances.

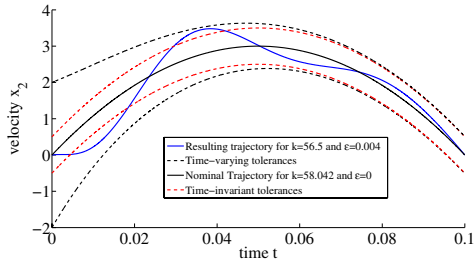


Figure 8. Modified tolerances.

7 CONCLUSION AND FUTURE RESEARCH

In this paper the robustness of flatness based tracking controllers using only output feedback for a magnetic levitation system has been analyzed. Verified integration of subsets of the uncertain parameter interval led to guaranteed enclosures of the admissible sets of parameters which lead to trajectories within the specified tolerances The algorithm can be applied directly to systems with uncertain initial conditions. It can also be extended to other control strategies. Future research will include the optimization of splitting and selection strategies and the implementation of other validated ODE solvers like VNODE [15], VALENCIA-IVP [2] or VSPODE [14].

References

[1] F. Antritter, M. Kletting, and E. P. Hofer. Robustness analysis of flatness based tracking controllers using interval methods. *Int. J. Control (To Appear)*, 2007.

[2] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SmartMOBILE: Improvement of Performance by ValEncIA-IVP. *accepted for Dagstuhl Seminar Proceedings 06021 Reliable Implementation of Real Number Algorithms: Theory and Practice, to appear in Springer LNCS.*

[3] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. Reliable Computing*, 4:361–369, 1998.*

[4] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Flatness and defect of nonlinear systems: introductory theory and examples. Int. J. Control*, 61:1327–1361, 1995.*

[5] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems. Trans. Aut. Control*, 44:922–937, 1999.*

[6] M. Fliess and J. Rudolph. Local tracking observers for flat systems. Proceedings of the Symposium on Control, Optimization and Supervision, CESA '96 IMACS Multiconference, Lille, France*, pages 213–217, 1996.*

[7] E. Freund. Zeitvariable Mehrgrößensysteme*. Lecture notes in operations and mathematical science 57, Springer-Verlag, New York, 1971.*

[8] V. Hagenmeyer and E. Delaleau. Exact feedforward linearisation based on differential flatness: The siso case. In Nonlinear and Adaptive Control (NCN4 2001), Lecture notes in Control and Information Sciences*, volume 281, pages 161–170. Springer, Berlin, Heidelberg, 2001.*

[9] V. Hagenmeyer and E. Delaleau. Exact feedforward linearization based on differential flatness. Int. J. Control*, 76:537–556, 2003.*

[10] M. B. K. Makino. Suppression of the Wrapping Effect by Taylor Model-based Verified Integrators: Long-term Stabilization by Preconditioning . International Journal of Differential Equations and Applications (2006, in print).

[11] M. B. K. Makino. Suppression of the Wrapping Effect by Taylor Model-based Verified Integrators: Long-term Stabilization by Shrink Wrapping. International Journal of Differential Equations and Applications (2006, in print).

[12] M. Kletting, A. Rauh, H. Aschemann, and E. P. Hofer. Interval Observer Design Based on Taylor Models for Nonlinear Uncertain Continuous-Time Systems. 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006, Duisburg, Germany, Book of abstracts, pp. 101-102, 2006.

[13] J. Levine, J. Lottin, and J. C. Ponsart. A nonlinear approach to the control of magnetic bearings. IEEE Trans. on Control Systems Technology*, pages 545 – 552, 1996.*

[14] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization for Dynamic Systems Using Interval Analysis. Presented at 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006, Duisburg, Germany, 2006.

[15] N. S. Nedialkov and K. R. Jackson. Methods for Initial Value Problems for Ordinary Differential Equations. In U. Kulisch, R. Lohner and A. Facius eds., Perspectives on Enclosure Methods, pages 219-264, Springer-Verlag, Vienna, 2001.

[16] H. Sira-Ramirez and S. K. Agrawal. Differentially Flat Systems*. Marcel Dekker, New York, 2004.*

Interval Observer Design Based on Taylor Models for Nonlinear Uncertain Continuous-Time Systems

Marco Kletting, Andreas Rauh, Eberhard P. Hofer
Institute of Measurement, Control, and Microtechnology
University of Ulm
D-89081 Ulm, Germany
{Marco.Kletting, Andreas.Rauh, EP.Hofer}@uni-ulm.de

Harald Aschemann
Chair of Mechatronics
University of Rostock
D-18059 Rostock, Germany
Harald.Aschemann@uni-rostock.de

Abstract

In most applications in control engineering not all state variables can be measured. Consequently, state estimation is performed to reconstruct the non-measurable states taking into account both system dynamics and the measurement model. If the system is subject to interval bounded uncertainties, an interval observer provides a guaranteed estimation of all states. The estimation consists of a recursive application of prediction and correction steps.

The prediction step corresponds to a verified integration of the system model describing the system dynamics between two points of time at which measured data is available. In this paper, a Taylor model based integrator is used.

Considering the state enclosures obtained in the prediction step, the correction step reconstructs states and parameters from the uncertain measurements with the help of a measurement model. The enclosures of states and parameters determined by the interval observer are consistent with both system and measurement models as well as all uncertainties.

1 INTRODUCTION

Interval methods are powerful tools for the calculation of guaranteed bounds for the state variables of systems with interval bounded uncertainties. In most applications in control engineering, a measurement of all states is either impossible or avoided because of reasons of sensor reliability and cost-effectiveness. Especially in nonlinear control, also the knowledge of non-measured system state variables is required. Therefore, state observers are employed to compute estimates for the whole state vector. If the system is influenced by interval uncertainties, an interval observer [8, 11, 15] determines guaranteed enclosures of all state variables and parameters. The observation consists of the recursive application of two steps and involves a system model describing the system dynamics and a measurement equation describing the sensor characteristics. The first step is the so-called prediction step, and the second step is denoted as correction step. After the extended state vector, consisting of state vector and the vector of uncertain parameters, are predicted by the uncertain system model, the extended state vector is estimated from the measured values in the correction step. The correction step is initialized with the result of the prediciton step. The prerequisite is that the system model, the measurement model, and all uncertainties provide guaranteed enclosures. In [5] an interval observer for cooperative systems has been proposed, where the state variables are enclosed by a single interval box. The approach in [6] is based on a branch and bound algorithm and the sets describing the state enclosure are given by multiple interval boxes. In [9] also an interval observer for cooperative systems has been proposed, in contrast to [5] multiple interval boxes are used for the state enclosures. The interval observer, which is proposed in this paper involves a Taylor model based integration [3, 13, 14] of the system dynamics in the prediction step. The state enclosures are given by Taylor models consisting of a multivariate polynomial and an interval remainder term. For a tight enclosure of non-convex sets with interval boxes often many boxes

are required. In case of Taylor models often a single Taylor model is sufficient to obtain the same enclosure quality. One of the advantages of interval boxes is that operations like the computation of the intersection of two intervals are easily performed, which is especially important for the implementation of the correction step. To determine a tight enclosure of a set describing the intersection of a Taylor model and an interval is however more difficult, thus the implementation of the correction step becomes more sophisticated. The paper is organized as follows: In Section 2, the problem formulation is given. In Section 3, the algorithm of the interval observer is presented in detail. The performance of the interval observer is demonstrated by simulation results in Section 4. Finally, conclusions and an outlook on future research are given in Section 5.

2 PROBLEM FORMULATION

Consider the following nonlinear uncertain system

$$\dot{x}(t) = f_x(x(t), p(t), t), \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector and $p \in \mathbb{R}^{n_p}$ the vector of the uncertain parameters. The parameter vector p and the initial conditions $x(0)$ are assumed to be uncertain with $p \in [\underline{p}; \overline{p}]$ and $x(0) \in [\underline{x}(0); \overline{x}(0)]$. If the parameters vary over time within their bounds and if upper and lower bounds of their variation rates are known, then

$$\dot{p}(t) = \Delta p \quad \text{with} \quad \Delta p \in [\Delta \underline{p}; \Delta \overline{p}] \tag{2}$$

holds. The state vector is extended by the parameter vector and the time variable t according to

$$z(t) = [x(t)^T, p(t)^T, t]^T \tag{3}$$

leading to an extended state space representation

$$\begin{aligned} \dot{z}(t) =& f(z(t)) \\ \text{with} \quad f(z(t)) =& \begin{bmatrix} f_x(x(t), p(t), t) \\ \Delta p \\ 1 \end{bmatrix} \end{aligned} \tag{4}$$

with $f : D \mapsto \mathbb{R}^n$, $D \subset \mathbb{R}^n = \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^1$. Components p_i of the vector of uncertain parameters which are time-invariant are characterized by $\Delta p_i = 0$.

In addition to the system dynamics, the interval observer requires a sensor model that describes the relation between the measurements and the true system states. The measurement equation is then given by

$$\begin{aligned} y\left(t_{k+1}\right) =& \tilde{h}\left(z\left(t\right), q\left(t\right)\right)\Big|_{t=t_{k+1}} + \delta\left(t\right)\Big|_{t=t_{k+1}} \\ =& h\left(z\left(t\right), q\left(t\right), \delta\left(t\right)\right)\Big|_{t=t_{k+1}} \end{aligned} \tag{5}$$

with $h : D \mapsto \mathbb{R}^m$, $D \subset \mathbb{R}^n \times \mathbb{R}^{n_q} \times \mathbb{R}^m$. The variable t_{k+1} denotes points of time with measurement information. The parameter vector $q \in \mathbb{R}^{n_q}$ of the measurement equation is bounded by $q \in [\underline{q}, \overline{q}]$. The additive measurement error $\delta \in \mathbb{R}^m$ is bounded by $\delta \in [\underline{\delta}, \overline{\delta}]$.

An interval observer determines a guaranteed enclosure of the extended state vector consting of the system states and the uncertain parameters. The basic concept of the interval observer is illustrated in Fig. 1. In general, the parameters q can be estimated by the interval observer as well.

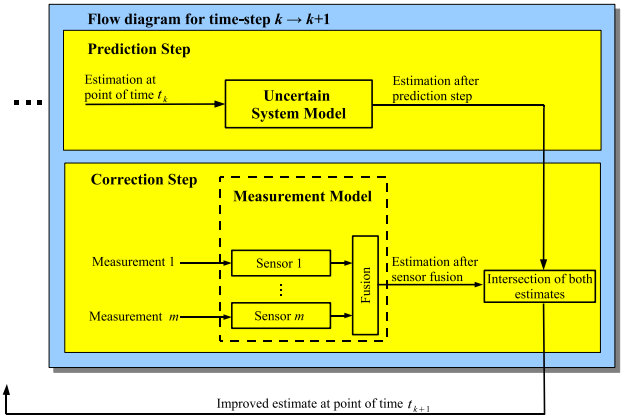


Figure 1. Flow diagram of the interval observer.

At a given point of time t_k the set describing the enclosure of extended state vector is predicted by a verified integration of the system model up to t_{k+1}. At t_{k+1}, the state variables are reconstructed from the uncertain measurements in the correction step based on the measurement model. The correction step is initialized with the enclosure of the extended state vector computed in the prediction step. Therefore, an improved enclosure is obtained. This procedure is repeated recursively. The prediction is performed until the next measurements are available, then the next correction step is applied.

3 INTERVAL OBSERVER BASED ON TAYLOR MODELS

3.1 Prediction Step

In the prediction step, a verified integration of the system model is performed. In this work, a Taylor model based integrator [3, 13, 14] is used which is described briefly in the following.

3.1.1 Basic Algorithm

The goal is to derive a Taylor model [3] consisting of a Taylor polynomial P and bounds for an interval remainder Term I. This interval remainder is the remainder error for the flow of the differential equation

$$\dot{z}(t) = f(z(t)) \tag{6}$$

in the time interval $[t_0; t_1]$ with a given solution at $z(t_0)$,where $z(t_0)$ may be an interval box but also a Taylor model. The initial interval box at $t = 0$ can be expressed as a Taylor model

$$z(t=0) = c + Dz_0 \text{ with } z_{0,i} \in [-1;1] \ \ , i = 1 \ldots, n \ \ , \tag{7}$$

where c is the midpoint of $z(t = 0)$ and D is a diagonal matrix with $d_{i,i} = 0.5 \cdot \mathrm{diam}(z(t = 0))$. In contrast to algorithms such as those implemented in VNODE not only a series expansion in time t but also in the initial extended state vector z_0 is performed, to reduce the dependency problem and the wrapping effect. In each time-step, the set enclosing the flow of the differential equation is described by a Taylor model, which allows for a more flexible enclosure of non-convex sets.

The integral of the (6) is given by

$$z(t) = z(t_0) + \int_{t_0}^{t} f(z(\tau))d\tau \ \ . \tag{8}$$

From the integral form a fix-point equation

$$\mathcal{O}(z)(t) = z(t_0) + \int_{t_0}^{t} f(z(\tau))d\tau \tag{9}$$

is obtained. The goal is now to determine a Taylor model of order ρ for the flow of the differential equation, such that

$$\mathcal{O}(P_{\rho,z} + I_z) \subseteq P_{\rho,z} + I_z \tag{10}$$

is fulfilled. A more detailed description of the algorithm can be found in [3].

3.1.2 Reduction of the Wrapping Effect

As already mentioned, the expansion in initial conditions reduces the dependency problem and the wrapping effect. In order to limit the long-term growth of the truncation error and to further reduce overestimation the following strategies can be applied.

- Shrink Wrapping: The interval remainder is absorbed in the polynomial part [14].
- Preconditioning: The solution of the ODE is studied in a different coordinate system in order to minimize long-term error growth [13].
- State-space extension by the remainder term during simulation.
- Splitting of the remainder term during simulation.
- Splitting of the reference domain during simulation.

In the following, splitting of the reference domain into subintervals shall be discussed in more detail.

Consider a Taylor model $T(z_0)$ with the reference domain z_0. The reference domain can be split into subintervals $\tilde{z}_0$. For each subinterval a new Taylor model $T(\tilde{z}_0)$ is obtained. To determine the component in which the reference domain has to be split, a sensitivity analysis is performed. The component μ of the reference domain z_0 is chosen for splitting in which the Taylor model $T(z_0)$ is most sensitive. For that purpose, all $w_{i,j}$

$$\begin{aligned} w_{i,j} =& \mathrm{diam}(z_{0,i}) \left| \frac{\partial T_j(z_0)}{\partial z_{i,0}} \Big|_{z_0 = \mathrm{mid}(z_0)} \right| \\ & i = 1, \ldots, n, \ j = 1, \ldots, n. \end{aligned} \tag{11}$$

have to be calculated and the component μ is determined by

$$\mu = \underbrace{\mathrm{argmax}}_{j=1,\ldots,n} \left(\sum_{i=1}^{n} w_{i,j} \right) \ \ . \tag{12}$$

In equation (11), the results are the coefficients of the linear part of the Taylor model $T(z_0)$ multiplied by the factor 2, because $\mathrm{mid}(z_{0,i}) = 0$ and $\mathrm{diam}(z_{0,i}) = 2$ for $z_{0,i} \in [-1;1] \ \ , i = \{1 \ldots, n\}$. For numerical and implementation reasons it is advantageous to have the unit box $[-1, 1]^n$ as a reference domain in each integration step [14]. To obtain again the unit box as a reference domain, $\tilde{z}_0$ is expressed as a Taylor model according to

$$\begin{aligned} \tilde{z}_0 =& \tilde{T}(z_0) = \tilde{c} + \tilde{D}\, z_0 \\ & \text{with } z_{0,i} \in [-1;1] \ \ , i = 1 \ldots, n \ \ , \end{aligned} \tag{13}$$

where $\tilde{c}$ is the midpoint of $\tilde{z}_0$ and $\tilde{D}$ is a diagonal matrix with $\tilde{d}_{i,i} = 0.5 \cdot \mathrm{diam}(\tilde{z}_{i,0})$. The components of the vector of the original reference domain z_0 of $T(z_0)$ are replaced by the components of $\tilde{T}$ by substituting $\tilde{T}_i(z_0)$ for $z_{0,i}$, which results in a modified Taylor model $\hat{T}(z_0) = T(\tilde{T}(z_0))$. The scaling to the unit box is done after each splitting operation.

If the state enclosure is already given by several Taylor models, the most appropriate Taylor model for the splitting of the reference domain has to be selected. This is done by calculating the interval enclosure of each Taylor model and the corresponding pseudo volume, which is the product of the diameters of the components of the resulting interval vector. The Taylor model with the largest pseudo volume is selected for splitting.

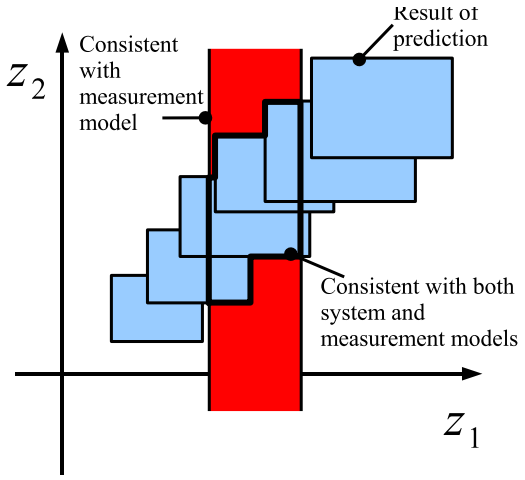


Figure 2. Interval observer based on subintervals.

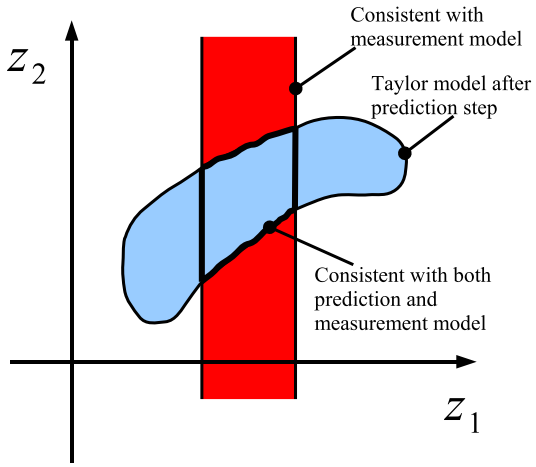


Figure 3. Interval observer based on Taylor models.

3.2 Correction Step

In the correction step, guaranteed enclosures of the state variables are reconstructed from the measurements and intersected with the results from the prediction step. In general, the relation between the measurements and the state variables is nonlinear, see equation (5).

If a state variable can be measured directly, the implementation of the correction step is fairly easy, if an interval observer based on subintervals [8, 11] is used. Then only, an intersection of the predicted result with the difference of the measured value and the measurement uncertainty is required. This is illustrated in Fig. 2. However, if an interval observer based on Taylor models is considered, even in this case, the correction step is not trivial, as depicted in Fig. 3.

3.2.1 Basic Correction Step

In order to perform the correction step, the Taylor models $T(z_0)$ of the state variables are subsitituted for z in the measurement equation

$$\begin{aligned} y(t_{k+1}) =& h\left(z\left(t\right), q\left(t\right), \delta\left(t\right)\right)\Big|_{t=t_{k+1}} \\ =& h\left(T(z_0), q\left(t\right), \delta\left(t\right)\right)\Big|_{t=t_{k+1}} . \end{aligned} \tag{14}$$

The right hand side of this equation represents again a Taylor model. Next, the measurement equation is rewritten according to

$$h\left(T(z_0), q\left(t\right), \delta\left(t\right)\right)\Big|_{t=t_{k+1}} - y(t_{k+1}) = 0 \ . \tag{15}$$

Equation (15) is solved for z_0 with an interval Newton method [7, 10], which leads to a tightened reference domain $\tilde{z}_0 \subseteq z_0$. For $n > m$, equation (15) is underdetermined and cannot be inverted directly. One solution approach is to solve the equation (15) only for m variables while the remaining $n-m$ variables are considered to be constant intervals. Another possibility is to consider a sufficient number of previous measurements $y(t \leq t_k)$ and the corresponding Taylor models of the right hand side of equation (14) to obtain the missing $n-m$ equations.

The components of the vector of the original reference conditions z_0 are replaced by the Taylor model of $\tilde{z}_0$ with $z_{0,i} \in [-1, 1]$ as reference domain as mentioned in Section 3.1.

If in the prediction step splitting of the reference domain has been applied and the predicted set is therefore represented by several Taylor models first the measurement equation is evaluated for all Taylor models. If for the interval bound B of the Taylor model resulting from $\tilde{h}(T(z_0), q(t))\Big|_{t=t_{k+1}}$

$$B\left(\tilde{h}(T(z_0), q(t))\Big|_{t=t_{k+1}}\right) \cap \Big(y(t_{k+1}) - \delta(t)\Big)\Big|_{t=t_{k+1}} = \emptyset \tag{16}$$

holds, the corresponding Taylor model is inconsistent with the measurement and the measurement errors. Therefore, it is deleted.

This approach can also be used for consideration of time-varying interval bounded parameters with interval bounded variation rates, as described in Section 2.1. The upper and lower bounds of the parameters can be considered as bounds for a measurement $y_p(t) \in [\underline{p}, \overline{p}]$ for all t. If these parameter

bounds are exceeded during the simulation, i.e. $\inf(p(t)) < \underline{p}$ or $\sup(p(t)) > \overline{p}$ for any t, they can be limited by the same approach that is implemented for the correction step of the observer.

3.2.2 Consistency Tests

To improve the correction step, consistency tests can be performed additionally. The reference domain z_0 is split into subintervals $\hat{z}_0$ and the measurement equation (5) is rewritten according to

$$\tilde{h}\left(z\left(t\right), q\left(t\right)\right)\Big|_{t=t_{k+1}} = y(t_{k+1}) - \delta(t)\Big|_{t=t_{k+1}} \quad . \tag{17}$$

Next, consistency tests are performed by evaluation of equation (17) for all Taylor models $T(\tilde{z}_0)$. Now, three different cases have to be distinguished:

1. If $B\left(\tilde{h}\left(T(\tilde{z}_0), q\left(t\right)\right)\Big|_{t=t_{k+1}}\right) \subseteq \left(y_{k+1} - \delta_{k+1}\right)$ holds, then $\hat{z}_0$ is consistent.
2. If $B\left(\tilde{h}\left(T(\tilde{z}_0), q\left(t\right)\right)\Big|_{t=t_{k+1}}\right) \cap \left(y_{k+1} - \delta_{k+1}\right) = \emptyset$ holds, then $\hat{z}_0$ is inconsistent and can be deleted.
3. All remaining subintervals $\tilde{z}_0$ have to be split further.

The obtained subset of the reference domain consists of several subintervals and for each subinterval Taylor models are obtained. In order to avoid an exponential growth of the number of Taylor models during simulation time, efficient merging strategies have to be applied. Subintervals can be merged in case of small overestimation of the union of the merged subintervals [16]. Another possibility is to replace the result by a single interval box which encloses all subintervals. This is a special case of the previously stated merging routine. In this work, another approach is proposed. The goal is to enclose the obtained set by a rotated box. The algorithm consists of the following steps:

1. Determine all interval vertices and midpoints .
2. Calculate the balance point c and the covariance matrix V of the distribution of these points.
3. Determine the eigenvectors of the covariance matrix.
4. The initial enclosure is determined by the balance point c and the and the eigenvectors of the covariance matrix V and can be expressed as a Taylor model

   $$T(z_0) = c + V\, z_0 \text{ with } z_{0,i} \in [-1; 1], \;\; i = 1 \ldots, n \;\; , \tag{18}$$

   where c corresponds to the balance point. V is a matrix containing the eigenvectors normalized to length 1 and z_0 is the unit interval box.
5. Check whether all subintervals are included in the initial enclosure. This is done by transforming all subintervals $\tilde{z}_0$ to $\tilde{z}_0^{'}$ by a subtraction of c and multiplication with V^{-1}, i.e.,

   $$\tilde{z}_0^{'} = V^{-1}(\tilde{z}_0 - c) \;\; . \tag{19}$$

   If not all $\tilde{z}_0^{'}$ are contained in the unit interval box $[-1, 1]^n$, the initial enclosure does not contain all intervals and has to be inflated until all intervals are contained.
6. If all intervals are contained, i.e.,

   $$V^{-1}(\tilde{z}_0 - c) \subseteq [-1, 1]^n \text{ for all } \tilde{z}_0 \;\; , \tag{20}$$

   a contraction is performed as long as significant further improvements can be achieved.
7. This enclosure or rotated box can be expressed again as a Taylor model according to

   $$\tilde{T}(z_0) = c + \tilde{V}\, z_0 \text{ with } z_{0,i} \in [-1; 1], \;\; i = 1 \ldots, n \;\; . \tag{21}$$

8. The components $z_{0,i}$ of the vector of the original initial conditions z_0 are replaced with the components of $\tilde{T}(z_0)$ by substituting $\tilde{T}_i(z_0)$ for $z_{0,i}$ which results in a modified Taylor model $\hat{T}(z_0) = T(\tilde{T}(z_0))$.

If the set of subintervals describing the reference domain after the consistency test is highly non-convex, it has to be approximated by several rotated boxes. This will be considered in future research.

4 SIMULATION RESULTS

In this Section, estimation results for two nonlinear systems are shown. In the first example, the algorithm with the basic correction step as described in Section 3.2.1 is applied. In the second example consistency tests are performed additionally.

4.1 Non-Isothermal Stirred Tank Reactor

Consider a jacketed non-isothermal stirred tank reactor [4], in which the van der Vusse reaction is taking place. The reaction kinetics are described by

$$\begin{aligned} A \overset{k_1}{\rightarrow} B \overset{k_2}{\rightarrow} C \text{ and} \\ 2A \overset{k_3}{\rightarrow} D. \end{aligned} \tag{22}$$

Component A represents the reactant cyclopentadine, B is the desired product cyclopentenol, component C is the unwanted side product cyclopentanediol, and D is dicyclopentadine, the product of the undesirable parallel reaction. The

concentration of the reactant A is given by c_a. The concentration of the component B is given by c_b. The temperature in the reactor is denoted by ν, and the jacket temperature by ν_K. The system is modeled by a set of four nonlinear differential equations

$$\begin{aligned}
\dot{c}_a &= D(c_{a0} - c_a) - k_1 c_a - k_3 c_a^2 \,, \\
\dot{c}_b &= -D c_b + k_1 c_a - k_2 c_b \,, \\
\dot{\nu} &= D(\nu_0 - \nu) + \frac{k_w A_r}{\rho C_p V_R}(\nu_K - \nu) \\
&\quad - \frac{1}{\rho C_p}(k_1 c_a \Delta H_{rab} + k_2 c_b \Delta H_{rbc} + k_3 c_a^2 \Delta H_{rad}) \,, \\
\dot{\nu_K} &= \frac{1}{m_k C_{pk}}(\dot{Q}_k + k_w A_R(\nu - \nu_K))
\end{aligned} \tag{23}$$

with $c_a(0) \in [0.25; 0.75]$mol/l, $c_b(0) \in [0.25; 0.75]$mol/l, $\nu(0) \in [100; 110]°C$, $\nu_K(0) \in [100; 110]°C$. The coefficients k_i, $i = \{1, 2, 3\}$, of the reaction kinetics are temperature-dependent according to

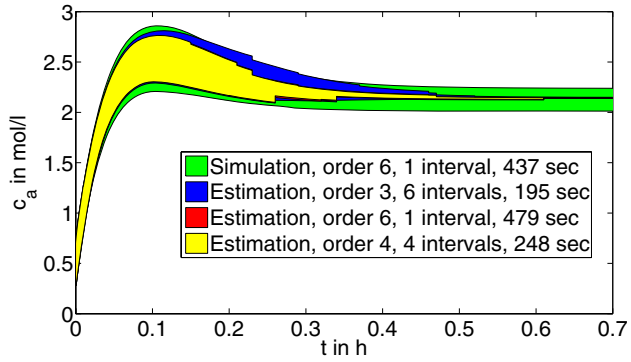
$$k_i(\nu) = k_{i,0} e^{-\frac{E_i}{\nu + 273.15}} \,. \tag{24}$$

The time-invariant parameter E_1 is assumed to be uncertain with $E_1 \in [-1.01; -0.99] \cdot 9758.3$K, while the values of E_2 and E_3 are assumed to be known exactly. All nominal parameter values can be found in [4].

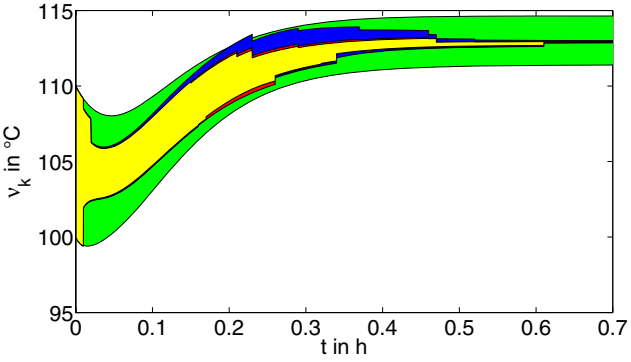
The measurement equation for the temperature in the reactor is given by

$$y(t_{k+1}) = \nu(t)\Big|_{t=t_{k+1}} + \delta(t)\Big|_{t=t_{k+1}} \,. \tag{25}$$

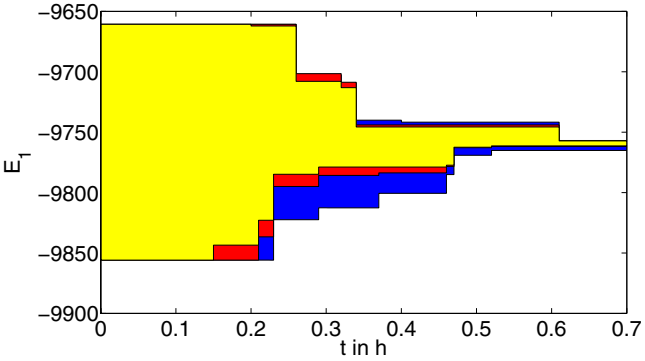
The measurement uncertainty $\delta(t)$ is assumed to be $[-1; 1]°$C for all t and it is further assumed that measurements are available every 0.01h. The measured values have been generated by a verified simulation of the system model with nominal values of the initial state vector and parameter E_1. A uniformly distributed noise, which was bounded by $\delta(t) \in [-1; 1]$, was added to the resulting values for $\nu(t)$. The simulation results for three different orders of the Taylor models and three different numbers of subintervals are shown. For all three simulations QR preconditioning has been chosen. In Figs. 4 (a) – 4 (c) the estimation results for c_a, v_k, and E_1 are compared. In Figs. 4 (a) and 4 (b) the result of a simulation without inclusion of measurement information is also included for comparison. Despite large uncertainties in the initial conditions of the state variables c_a,v_k, and also the uncertain parameter E_1, the intervals are tightened significantly already after a few time-steps. It can be seen that c_a is very sensitive with respect to E_1 as it tightens significantly as soon as a large range of E_1 is excluded by the interval observer. The most conservative results stem from the simulation of order 3 in time and in the initial state



(a) Time-dependent concentration c_a.



(b) Time-dependent temperature ν_K.



(c) Estimation of the uncertain time-invariant parameter E_1.

Figure 4. Estimation results.

vector and with 6 subintervals. Increasing to the order 6 without interval splitting improves the results, but the computating time is more than two times longer. The tightest results have been achieved for order 4 with 4 subintervals, where the computing time was between the previous two. The results show that only increasing the order does not always lead to the tightest enclosures. A combination with interval splitting is more efficient concerning both simulation quality and computation time.

4.2 Isothermal Stirred Tank Reactor

An isothermal gas-phase reactor [1] is charged with an initial amount of gaseous reactands A and B, in which both substances react according to the reversible reaction kinetics

$$2A \rightleftharpoons B \ . \tag{26}$$

The system model which considers the partial pressures z_1 of component A and z_2 of component B is given by

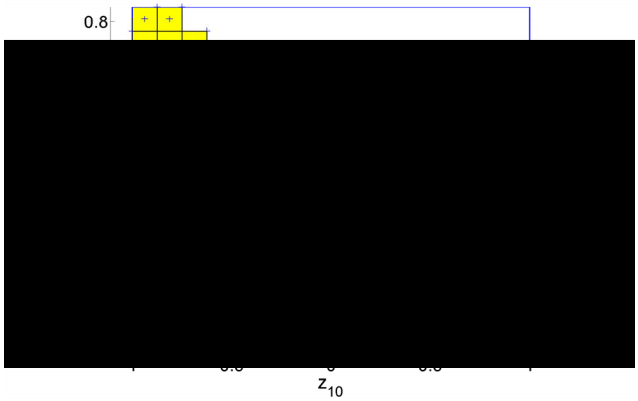
$$\begin{aligned} \dot{z}_1(t) &= -2k_1 z_1^2 + 2k_2 z_2 \\ \dot{z}_2(t) &= k_1 z_1^2 - k_2 z_2 \end{aligned} \tag{27}$$

with $z_1(0) \in [0; 5]$ bar and $z_2(0) \in [0.5; 1.5]$ bar.The total pressure can be measured. The measurement equation is given by

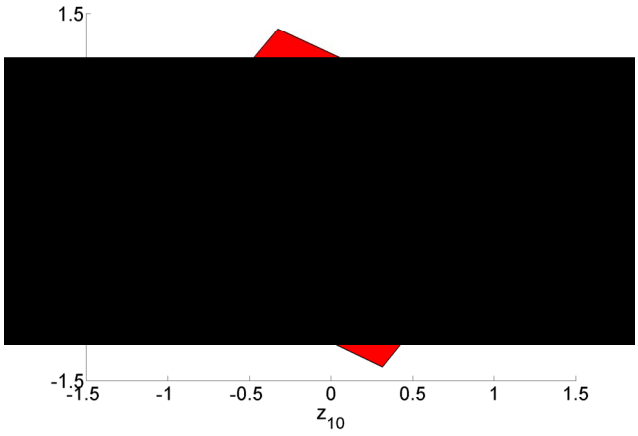
$$y(t_{k+1}) = z_1(t)\Big|_{t=t_{k+1}} + z_2(t)\Big|_{t=t_{k+1}} + \delta(t)\Big|_{t=t_{k+1}} \ . \tag{28}$$

The additive measurement uncertainty is given by $\delta \in [-0.1; 0.1]$ bar for all t. A fixed step-size $T = 2.5$ s was used. It is assumed that only a single measurement at $t = 200$ s is available.

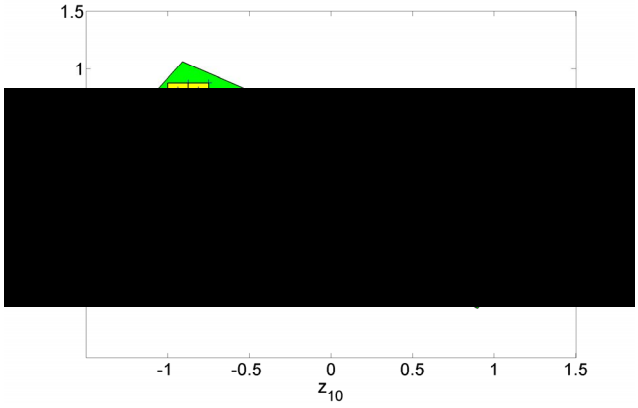
Three different simulations have been performed. One without consistency tests in the correction step (case A), one with consistency tests and replacement of the resulting subintervals of the reference domain by a single interval box (case B), and one by replacement of the resulting subintervals by a rotated box (case C). The order of the Taylor models in time and in the initial state vector was 5 in all simulations. Additionally, QR-preconditioning has been applied. Fig. 5(a) shows the remaining subintervals of the reference domain after the consistency test together with the interval vertices, midpoints, and the balance point. Fig. 5(b) depicts the corresponding initial enclosure and Fig. 5(c) the final enclosure, which is used as a new reference domain replacing the original reference domain z_0 by the corresponding Taylor model of the rotated box. The simulation that employs only the basic correction step (case A) leads to the most conservative enclosures. The results with consistency test and replacement of the remaining boxes by one interval box (case B) are slightly tighter. For the enclosure of z_2, significant improvement is achieved if the rotated box



(a) Result of the consistency tests.



(b) Initial enclosure.



(c) Final enclosure.

Figure 5. Inclusion by rotated box.

is used as an approximation of the remaining interval boxes (case C).

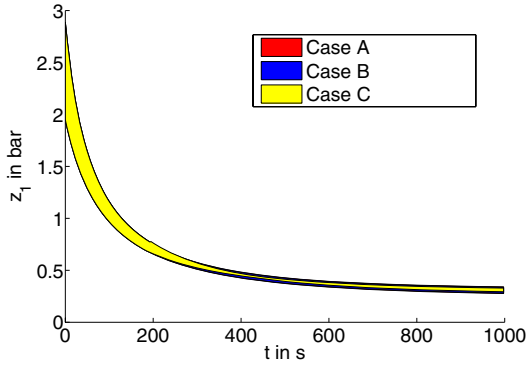


Figure 6. Time-dependent enclosures of z_1.

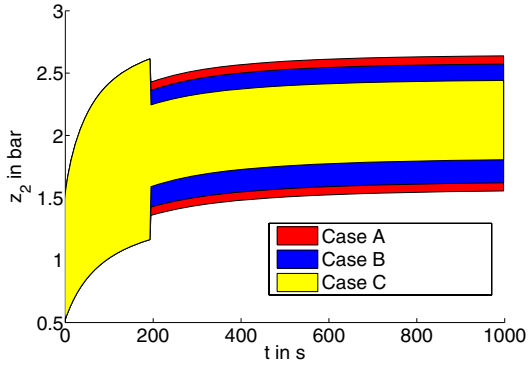


Figure 7. Time-dependent enclosures of z_2.

5 CONCLUSIONS

In this paper, an interval observer based on Taylor models for guaranteed state and parameter estimation for nonlinear uncertain continuous-time systems has been presented. The interval observer calculates those regions of the extended state vector, which consist of the system state variables and the uncertain parameters, that are consistent with the system model, the measurement model, the measurements, and all uncertainties. Two applications have been discussed in order to illustrate the performance of the algorithm. It has been shown how an efficient correction step can be implemented, when the sets are described by one or several Taylor models. Further research will concentrate on the optimization of the correction step, merging and reapproximation routines, and a comparison of interval observers based on other verified ODE solvers, e.g. VNODE, VALENCIA-IVP [2], or VSPODE [12]. Concerning the last point, it is expected that the performance of each interval observer strongly depends on the considered system.

Acknowledgements

The authors would like to thank Martin Berz and Kyoko Makino for making the COSY VI integrator available.

References

[1] T. Alamo, J. Bravo, and E. F. Camacho. Guaranteed State Estimation by Zonotopes. In *Proc. of the 42nd IEEE Conference on Decision and Control, Maui, Hawai USA, December 2003, pp. 5831–5836.*

[2] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SMARTMOBILE: Improvement of Performance by VALENCIA-IVP. Proc. of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice, Lecture Notes in Computer Science, 2006,(in print).

[3] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. *Reliable Computing*, 4:361–369, 1998.

[4] S. Engell. *Entwurf nichtlinearer Regelungen*. Oldenbourg Verlag, Munich, Germany, 1995 (in German).

[5] J. L. Gouze, A. Rapaport, and Z. M. Hadj-Sadok. Robustness analysis of flatness based tracking controllers using interval methods. *Int. J. Control,2000, 133:45-56.*

[6] L. Jaulin. Nonlinear Bounded-Error State Estimation of Continuous-Time Systems. *Automatica*, 38:1079–1082, 2002.

[7] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer-Verlag, London, Great Britain, 2001.

[8] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed Recursive Nonlinear State Bounding Using Interval Analysis. *International Journal of Adaptative Control and Signal Processing*, 6(3):193–218, 2002.

[9] M. Kieffer and E. Walter. Nonlinear Parameter and State Estimation for Cooperative Systems in a Bounded-Error Context. Proc. International Dagstuhl Seminar: Software with Result Verification. Revised Papers edited by R. Alt, A. Frommer, R.B. Kearfott, W. Luther, Springer, Lecture Notes in Computer Science 2991, 2004, pp. 107-123.

[10] M. Kletting, A. Rauh, H. Aschemann, and E. P. Hofer. Consistency Techniques for Simulation of Wastewater Treatment Processes with Uncertainties. In *DVD Proc. of IFAC World Congress 2005*, Prague, Czech Republic, 2005.

[11] M. Kletting, A. Rauh, H. Aschemann, and E. P. Hofer. Interval Observer Design for Nonlinear Systems with Uncertain Time-Varying Parameters. In *Proc. of 12th IEEE Intl. Conference on Methods and Models in Automation and Robotics MMAR*, pages 361–366, Miedzyzdroje, Poland, 2006.

[12] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization for Dynamic Systems Using Interval Analysis. 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006, Duisburg, Germany, Book of abstracts, p. 74, 2006.

[13] K. Makino and M. Berz. Suppression of the Wrapping Effect by Taylor Model-Based Verified Integrators: Long-Term Stabilization by Preconditioning. *International Journal of Differential Equations and Applications (2006, in print)*.

[14] K. Makino and M. Berz. Suppression of the Wrapping Effect by Taylor Model-Based Verified Integrators: Long-Term Stabilization by Shrink Wrapping. *International Journal of Differential Equations and Applications (2006, in print)*.

[15] T. Raissi, N. Ramdani, and Y. Candau. Set Membership State and Parameter Estimation for Systems Described by Nonlinear Differential Equations. *Automatica 40 (2004) 1771-1777*.

[16] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Reduction of Overestimation in Interval Arithmetic Simulation of Biological Wastewater Treatment Processes. *Journal of Computational and Applied Mathematics*, 199(2):207–212, 2007.

Strong Unboundedness of Interval Linear Programming Problems

Jana Koníčková
Czech Technical University in Prague
Faculty of Civil Engineering
Department of Mathematics
Thákurova 7, 166 29 Praha 6, Czech Republic
konicko@fsv.cvut.cz

Abstract

A linear programming problem whose coefficients are prescribed by intervals is called strongly unbounded if each linear programming problem obtained by fixing coefficients in these intervals is unbounded. In the main result of this paper a necessary and sufficient condition for strong unboundedness of an interval linear programming problem is described. In order to have a full picture we also show conditions for strong feasibility and strong solvability of this problem. The necessary and sufficient conditions for strong feasibility, strong solvability and strong unboundedness can be verified by checking the appropriate properties by the finite algorithms. Checking strong feasibility and checking strong solvability are NP-hard. We show that checking strong unboundedness is NP-hard as well.

1. Introduction

We study an *interval linear programming (ILP)* problem

$$\max\{c^T x; Ax = b, x \geq 0\}, \tag{1}$$
$$A \in A^I, b \in b^I, c \in c^I, \tag{2}$$

where $A \in \mathrm{R}^{m\times n}$, $b \in \mathrm{R}^m$, $c \in \mathrm{R}^n$, A^I is an interval matrix, b^I and c^I are interval vectors. The interval matrix A^I is defined by

$$A^I = \{A; \underline{A} \leq A \leq \overline{A}\} = [\underline{A}, \overline{A}] = [\check{A} - \Delta, \check{A} + \Delta],$$

where $\check{A} = (\underline{A} + \overline{A})/2$ is the center matrix of A^I, $\Delta = (\overline{A} - \underline{A})/2$ is the radius matrix of A^I.
The interval vectors b^I, c^I are defined analogously:

$$\begin{aligned} b^I &= \{b; \underline{b} \leq b \leq \overline{b}\} = [\underline{b}, \overline{b}] = [\check{b} - \delta, \check{b} + \delta], \\ c^I &= \{c; \underline{c} \leq c \leq \overline{c}\} = [\underline{c}, \overline{c}] = [\check{c} - \gamma, \check{c} + \gamma]. \end{aligned}$$

A system $A^I x = b^I$ is called *strongly feasible* if each system $Ax = b$ with data satisfying $A \in A^I$, $b \in b^I$ has a nonnegative solution. The ILP problem is called *strongly feasible* if system $A^I x = b^I$ is strongly feasible. The ILP problem is called *strongly solvable* if each problem (1) with data satisfying (2) has an optimal solution. The ILP problem is called *strongly unbounded* if each problem (1) with data satisfying (2) is unbounded.

The necessary and sufficient conditions for strong feasibility and strong solvability of the ILP problem are given by Rohn ([2], [5]). These problems are NP-hard (proved by Rohn [2], [6]). These results are mentioned in Section 2.

In this paper a necessary and sufficient condition for strong unboundedness of the ILP problem is given (Theorem 13). We show that checking strong unboundedness is NP-hard (Theorem 17). In Section 3 several results on weak solvability of interval linear inequalities are presented, which we use in the proof of Theorem 13.

We define

$$Y_m = \{y \in \mathbb{R}^m;\ y_j \in \{-1, 1\} \text{ for each } j\},$$

i.e., Y_m is the set of all ± 1 vectors in $\mathbb{R}^m$. Obviously, the cardinality of Y_m is 2^m. For each $y \in Y_m$ we denote

$$T_y = \mathrm{diag}(y_1, \ldots, y_m) = \begin{pmatrix} y_1 & 0 & \ldots & 0 \\ 0 & y_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & y_m \end{pmatrix}.$$

For each $x \in \mathbb{R}^m$ we define its sign vector $\mathrm{sgn}\, x$ by

$$(\mathrm{sgn}\, x)_i = \begin{cases} -1 & \text{if } x_i < 0, \\ 1 & \text{if } x_i \geq 0, \end{cases}$$

$(i = 1, \ldots, m)$; obviously, $\mathrm{sgn}\, x \in Y_m$. For the sign vector $z = \mathrm{sgn}\, x$ the relation $|x| = T_z x$ holds. For each $y \in Y_m$ we define

$$\begin{aligned} A_y &= \check{A} - T_y \Delta, \\ b_y &= \check{b} + T_y \delta. \end{aligned}$$

The definition implies that

$$(A_y)_{ij} = \begin{cases} \overline{A}_{ij} & \text{if } y_i = -1, \\ \underline{A}_{ij} & \text{if } y_i = 1, \end{cases}$$

$$(b_y)_{ij} = \begin{cases} \underline{b}_i & \text{if } y_i = -1, \\ \overline{b}_i & \text{if } y_i = 1, \end{cases}$$

$(i = 1, \ldots, m,\ j = 1, \ldots, n)$.

The subsystem $A_y x = b_y$ is called *extremal subsystem.* Then the i-th equation of this subsystem has either the form $(\overline{A}x)_i = \underline{b}_i$ (if $y_i = -1$), or the form $(\underline{A}x)_i = \overline{b}_i$ (if $y_i = 1$). Extremal subsystems $A_y x = b_y$ are used in finite characterizations of strong feasibility and strong unboundedness.

2. Strong feasibility, strong solvability of the ILP problem

In this section we show the necessary and sufficient conditions for strong feasibility and strong solvability of the ILP problem, and complexity results for these problems, which are proved by Rohn. In the strong feasibility condition the extremal subsystems are used.

Theorem 1 (Rohn [2], [5]) *A system $A^I x = b^I$ is strongly feasible if and only if for each $y \in Y_m$ the system*

$$A_y x = b_y$$

has a nonnegative solution.

Theorem 2 (Rohn [5]) *The ILP problem (1), (2) is strongly solvable if and only if the following conditions are satisfied:*

1. *The ILP problem is strongly feasible.*
2. *The problem $\max\{\overline{c}^T x; \underline{A}x \le \overline{b}, \overline{A}x \ge \underline{b}, x \ge 0\}$ has an optimal solution.*

Theorem 3 (Rohn [2], [6]) *Checking strong feasibility of interval linear equations is NP-hard.*

Theorem 4 (Rohn [2], [6]) *Checking strong solvability of an interval linear programming problem is NP-hard.*

3. Strong unboundedness of the ILP problem

In the proof of the main result we use the duality theory (see Dantzig [1], Vanderbei [7]). The dual problem of the primal ILP problem (1), (2) is

$$\min\{b^T p; A^T p \ge c\}, \tag{3}$$

$$A \in A^I, b \in b^I, c \in c^I. \tag{4}$$

The dual problem (3) is called *feasible* if $A^T p \ge c$ has a solution. A system $(A^I)^T p \ge c^I$ is called *strongly unsolvable* if there is no $A \in A^I,\ c \in c^I$ such that the system $A^T p \ge c$ has a solution. The dual problem (3), (4) is called *strongly infeasible* if the system $(A^I)^T p \ge c^I$ is strongly unsolvable.

A vector $p \in \mathbb{R}^m$ is called a *weak solution* of a system of interval linear inequalities $(A^I)^T p \ge c^I$ if it satisfies $A^T p \ge c$ for some $A \in A^I,\ c \in c^I$. A system $(A^I)^T p \ge c^I$ is called *weakly solvable* if some system $A^T p \ge c$, $A \in A^I,\ c \in c^I$ is solvable (i.e., if there exists a weak solution of a system $(A^I)^T p \ge c^I$). Gerlach [3] proved the following characterization of weak solution of $A^I x \le b^I$.

Theorem 5 (Gerlach 1981 [3], [2]) *A vector x is a weak solution of $A^I x \le b^I$ if and only if it satisfies*

$$\check{A}x - \Delta|x| \le \overline{b}. \tag{5}$$

The following theorem is an analogue of the Gerlach theorem for the case of reverse inequality.

Theorem 6 *A vector p is a weak solution of $(A^I)^T p \ge c^I$ if and only if it satisfies*

$$\check{A}^T p + \Delta^T |p| \ge \underline{c}. \tag{6}$$

PROOF. If p solves $A^T p \ge c$ for some $A \in A^I$, $c \in c^I$, then

$$\begin{aligned} \underline{c} &\le c \le A^T p = (\check{A} + (A - \check{A}))^T p = \\ &= \check{A}^T p + (A - \check{A})^T p \le \check{A}^T p + |A - \check{A}|^T |p| \le \\ &\le \check{A}^T p + \Delta^T |p|, \end{aligned} \tag{7}$$

which is (6). Conversely, let (6) hold for some vector p. We denote $z = \operatorname{sgn} p$, then $T_z p = |p|$ and substituting it into (6) we get

$$\begin{aligned} \check{A}^T p + \Delta^T |p| &= \check{A}^T p + \Delta^T T_z p = (\check{A}^T + \Delta^T T_z) p = \\ &= (\check{A} + T_z \Delta)^T p \ge \underline{c}. \end{aligned} \tag{8}$$

Obviously, $(\check{A} + T_z \Delta) \in A^I$, $\underline{c} \in c^I$, hence p is a weak solution of $(A^I)^T p \ge c^I$. □

Weak solvability of a system $A^I x \le b^I$ was characterized by Rohn, this result is based on the Gerlach theorem.

Theorem 7 (Rohn [2]) *A system $A^I x \le b^I$ is weakly solvable if and only if the system*

$$(\check{A} - \Delta T_z) x \le \overline{b} \tag{9}$$

is solvable for some $z \in Y_n$.

For a system with reverse inequality we have a similar condition.

Theorem 8 *A system $(A^I)^T p \geq c^I$ is weakly solvable if and only if the system*

$$A_y^T p \geq \underline{c} \tag{10}$$

is solvable for some $y \in Y_m$ ($A_y = \check{A} - T_y\Delta$).

PROOF. Let the system $(A^I)^T p \geq c^I$ be weakly solvable. Then there exists a vector $p \in \mathbb{R}^m$ which is a solution of the system $A^T p \geq c$ for some $A \in A^I$, $c \in c^I$, then vector p is a weak solution. Denote $z = \operatorname{sgn} p$, then $|p| = T_z p$. According to Theorem 6, and (8) in its proof, the inequality $(\check{A} + T_z\Delta)^T p \geq \underline{c}$ holds. We denote $y = -z$, obviously $T_z = -T_{-z}$, hence

$$\begin{aligned}(\check{A} + T_z\Delta)^T p &= (\check{A} - T_{-z}\Delta)^T p = \\ &= (\check{A} - T_y\Delta)^T p \geq \underline{c}, \end{aligned} \tag{11}$$

which is (10). Conversely, if p satisfies $A_y^T p \geq \underline{c}$ for some $y \in Y_m$, then it is a weak solution of $(A^I)^T p \geq c^I$ and this system is weakly solvable. □

As we have seen in the proof of Theorem 8, the weak solvability condition can be formulated in the following form.

Theorem 9 *A system $(A^I)^T p \geq c^I$ is weakly solvable if and only if the system*

$$(\check{A} + T_y\Delta)^T p \geq \underline{c} \tag{12}$$

is solvable for some $y \in Y_m$.

The following remark is the consequence of the proof of Theorem 8.

Remark 10 *If p is a weak solution of the system $(A^I)^T p \geq c^I$, then it satisfies $(\check{A} + T_y\Delta)^T p \geq \underline{c}$ for $y = \operatorname{sgn} p$, and $(\check{A} - T_y\Delta)^T p \geq \underline{c}$ for $y = -\operatorname{sgn} p$.*

For the proof of the main result we need the negated form of Theorem 8.

Theorem 11 *A system $(A^I)^T p \geq c^I$ is strongly unsolvable if and only if there is no $y \in Y_m$ such that the system*

$$A_y^T p \geq \underline{c} \tag{13}$$

is solvable.

The main result of this paper, the strong unboundedness condition, is the consequence of the following theorem.

Theorem 12 *Let the ILP problem (1), (2) be strongly feasible. Then the following assertions are equivalent.*

1. *For each $A \in A^I$, $b \in b^I$, $c \in c^I$ the problem $\max\{c^T x; Ax = b, x \geq 0\}$ is unbounded.*
2. *There is no $A \in A^I$, $c \in c^I$ such that the system $A^T p \geq c$ has a solution.*
3. *There is no $y \in Y_m$ such that the system $A_y^T p \geq \underline{c}$ has a solution.*
4. *For each $y \in Y_m$ the problem $\max\{\underline{c}^T x; A_y x = b_y, x \geq 0\}$ is unbounded.*

PROOF. We use the duality theory for the primal problem $\max\{c^T x; Ax = b, x \geq 0\}$ and the dual problem $\min\{b^T p; A^T p \geq c\}$. The unboundedness of the primal problem implies the infeasibility of the dual problem. If the dual problem is infeasible, then the primal problem is either unbounded or infeasible. Under assumption of strong feasibility of the primal ILP problem we obtain the equivalence of assertions 1 and 2 and also equivalence of assertions 3 and 4. The equivalence of assertions 2 and 3 holds according to Theorem 11. □

The equivalence of assertions 1 and 4 in Theorem 12 holds without the assumption of strong feasibility, consequently we get the main result:

Theorem 13 *The ILP problem (1), (2) is strongly unbounded if and only if for each $y \in Y_m$ the problem*

$$\max\{\underline{c}^T x; A_y x = b_y, x \geq 0\} \tag{14}$$

is unbounded.

PROOF. If the ILP problem is strongly unbounded, then it is strongly feasible and according to Theorem 12, its assertion 4 holds. Conversely, if condition 4 of Theorem 12 holds, then from the strong feasibility condition in Theorem 1 it follows that the ILP problem is strongly feasible. Hence the ILP problem is strongly unbounded. □

If we apply Theorem 13 to the interval linear programming problem $\max\{\sum_{i=1}^n x_i; Ax = b, x \geq 0\}, A \in A^I, b \in b^I$, we get the following condition for the sets of feasible solutions, given by Mráz [4]. We denote $X(A, b) = \{x; Ax = b, x \geq 0\}$.

Theorem 14 (Mráz [4]) *The set $X(A, b)$ is unbounded for each $A \in A^I$, $b \in b^I$ if and only if the set $X(A_y, b_y)$ is unbounded for each $y \in Y$.*

4. Complexity result

In this section we show the complexity result for checking strong unboundedness of the ILP problem. In the proof of this result we use the complexity result for interval linear inequalities, which was proved by Rohn.

Theorem 15 (Rohn [2]) *Checking weak solvability of interval linear inequalities is NP-hard.*

Consequently, the problem of checking whether the system of interval linear inequalities

$$(A^I)^T p \geq c^I$$

has at least one solution is NP-hard.

We need the negated form of Theorem 15.

Theorem 16 *The problem of checking whether there is no $A \in A^I$, $c \in c^I$ such that the system of linear inequalities $A^T p \geq c$ has a solution (strong unsolvability) is NP-hard.*

Finally, we have the following complexity result.

Theorem 17 *Checking strong unboundedness of an interval linear programming problem is NP-hard.*

PROOF. When the NP-hard problem of checking strong unsolvability of interval linear inequalities can be reduced in polynomial time to the problem of checking strong unboundedness of the ILP problem, then the second problem is NP-hard as well.

Consider the interval linear programming problem

$$\min\{0^T p; A^T p \geq c\}, \tag{15}$$
$$A \in A^I, c \in c^I. \tag{16}$$

Checking strong infeasibility of this problem is NP-hard (see Theorem 16 and the definition of strong infeasibility of the dual ILP problem (3), (4)). The problem (15), (16) is the dual problem for the primal interval linear programming problem

$$\max\{c^T x; Ax = 0, x \geq 0\}, \tag{17}$$
$$A \in A^I, c \in c^I. \tag{18}$$

For each $A \in A^I$, $c \in c^I$ the primal problem (17), (18) has the solution $x = 0$, hence this ILP problem is strongly feasible. Consequently, according to the duality theory, the primal problem (17), (18) is strongly unbounded if and only if the dual problem (15), (16) is strongly infeasible. Hence checking strong unboundedness of the ILP problem is NP-hard. □

5. Conclusion

Characterization of strong unboundedness and the complexity result for this problem complete the results on basic properties of an ILP problem, which are the characterization of strong feasibility, characterization of strong solvability, and the complexity results for these problems. The necessary and sufficient conditions for strong feasibility, strong solvability and strong unboundedness can be verified by the finite algorithms, these problems are NP-hard. Hence, unless the famous conjecture P $\neq$ NP is false, there does not exist a necessary and sufficient condition for checking respective property which could be verified in polynomial time.

Acknowledgement

This work was supported by the research project MSM 6840770001.

References

[1] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, 1963.

[2] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, and K. Zimmermann. *Linear Optimization Problems with Inexact Data*. Springer, New York, 2006.

[3] W. Gerlach. Zur Lösung linearer Ungleichungssysteme bei Störung der rechten Seite und der Koeffizientenmatrix. *Math. Operationsforsch. Statist. Ser. Optim.*, 12:41–43, 1981.

[4] F. Mráz. Nonnegative solutions of interval linear systems. In L. Atanassova and J. Herzberger, editors, *Computer Arithmetics and Enclosure Methods*, pages 299–308. Elsevier Science Publishers B. V. (North-Holland), 1992.

[5] J. Rohn. Strong solvability of interval linear programming problems. *Computing*, 26:79–82, 1981.

[6] J. Rohn. Linear programming with inexact data is NP-hard. *Zeitschrift für angewandte Mathematik und Mechanik*, 78, Supplement 3:S1051–S1052, 1998.

[7] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996.

intpakX - An Interval Arithmetic Package for Maple

Walter Krämer
University of Wuppertal
Scientific Computing / Software Engineering
42119 Wuppertal, Germany
kraemer@math.uni-wuppertal.de

Abstract

intpakX *is a Maple package for (multiple precision) interval arithmetic. It contains data types, basic arithmetic and standard functions for real interval arithmetic and complex disc arithmetic. Moreover, it implements a handful of algorithms for validated numerical computing and graphical output functions for the visualization of results. The package* intpakX *thus gives the user the opportunity to do validated computing with a Computer Algebra System.*

The package may be used to solve (set valued) numerical problems in a computer algebra environment with mathematical rigor. But it is also a valuable didactical tool which allows the illustration of various interval methods/algorithms.

The latest version of intpakX offers the possibility to compute and to visualize the exact solution set of two and three dimensional linear interval systems [3]. This new feature will be discussed in some detail.

1. Introduction

The first very basic intpak version was created in 1993 by R. Corless and A. Connell [2] as an effort to incorporate real intervals into Maple. In 1999, the extension intpakX was released by I. Geulig and W. Krämer [4] incorporating a range of applications, several visualization tools, and an additional part for complex (interval) numbers. Release intpakX v1.0 is a completely redesigned package, combining the formerly separate packages in one new version [5]. It has been released by Waterloo Maple as *Maple PowerTool Interval Arithmetic* [1]. Meanwhile, a further extension is available [3]. It allows the user to visualize exact solution sets of linear systems of equations with interval coefficients in two and in three dimensions.

The most important feature of the package is the introduction of new (multiple precision) data types into Maple for *real intervals* and *complex disc intervals*. A range of operators and applications for these data types (see below) have been implemented (with separate names), so that the new interval types do not rely on the (rough) standard Maple notion of an interval. Also rounding is done separately to provide the correct rounding mode as needed in interval computation. So, intpakX intervals can be used safely with the implemented operators.

The graphical functions included in intpakX make it more convenient to use Maple graphics for interval computations. They use Maple graphics features to offer special output for the visualization of the intervals resulting from the concerned intpakX functions.

As mentioned above, intpakX defines Maple types for real intervals and complex disc intervals.

On the level of basic operations, intpakX includes the four basic arithmetic operators, including extended interval division as an extra function. Furthermore there are power, square, square root, logarithm and exponential functions (note that square is implemented separately from general multiplication as needed for intervals), a set of standard functions and union and intersection. Reimplementations of the Maple construction, conversion and unapplication functions are added.

As applications, *Verified Computation of Zeros* (Interval Newton Method) with the possibility to find all zeros of a function on a specified interval, and *Range Enclosure* for real-valued functions of one or two variables are implemented, the latter using either interval evaluation or evaluation via the mean value form and adaptive subdivision of inter-

vals. The user can choose between a non-graphical and a graphical version of the above algorithms displaying the resulting intervals of each iteration step.

Additionally, there is a range of operators for complex disc arithmetic. Besides the basic arithmetic operators, there are area-optimal multiplication and division as an alternative to carry out these operations. As a further function, the complex exponential function has been implemented. Range enclosure for complex polynomials serves as an application for complex interval arithmetic.

The latest version of `intpakX` also offers the possibility to compute and to visualize the exact solution set of two and three dimensional linear interval systems. This extension of `intpakX` is the outcome of a Bachelor-Thesis written by Sven Braun [3].

The package `intpakX` is freely available on the web [16]. To get the source code of the Braun-package [3], write a mail to `kraemer@math.uni-wuppertal.de`.
How to install `intpakX` and how to use this Maple Power Tool in your own worksheets is described in the source files, which are publicly available on the web (see `http://www.math.uni-wuppertal.de/~xsc/software/intpakX/`). From this link you get the most recent version of `inpakX` (the version presented by Maplesoft may an older version). In addition, the preprint *Introduction to the Maple Power Tool* `intpakX`, available on the web [7]. Its intention is to be a primer to `intpakX`.

Please note: `intpakX` assumes that the basic operations provided by Maple are accurate to at least one unit in the last place (ulp) with respect to the actual value of Maple's environment variable `Digits` (number of decimal digits Mapel uses to represent software floating-point numbers). If so, it is guaranteed that `intpakX` computes enclosures for ranges of expressions build from these basic operations. Up to now, a violation of the assumption is not known by the authors of `intpakX`.

A similar statement (1 ulp accuracy) about the accuracy of Maple functions like exp, log, sin, Bessel, etc., is probably not correct. Therefore `intpakX` uses several guard digits in its computations. Nevertheless computation of expressions involving such functions can not be guaranteed to enclose the true range! Despite this limitation, we are convinced that `intpakX` is a valuable didactical tool to illustrate interval algorithms/methods. As soon as error bounds for Maple funcions will be available, `intpakX` will use the bounds to compute guaranteed enclosures.

Some critical remarks on the reliability of the computer algebra system Maple may be found in [8]. Examine the state of the art in automating mathematics critically. Current programs (Maple, Mathematica and so on) (may) have many flaws and shortcomings. Don't rely blindly on the outcomes of nontrivial mathematical software. The validity of `intpakX` results depends strongly on the correctness of the outcome of Maple commands. For example, the Interval Newton Method provided by `intpakX` depends on the correctness of Maple's differentiation facility when creating the derivative of the function under consideration. In the following example the mathematically correct value of the first derivative of the continuously differentiable function $f(x)$ at the point $x = 0$ is 1/2 but Maple computes the erroneous value 0:

```
>f:=proc(x)
    if x=0 then 1
    else (exp(x) - 1)/x
    end if
 end proc:
 D(f)(0); #first derivative at 0
0
```

We refer once more to [8].

Two applications using `intpakX` will be discussed in the following sections. The source code segments presented below should be self-explanatory to a large extend. End-of-line-comments start with #, and > indicates a Maple input line. To become deeper acquainted with `intpakX`'s commands, please consult [7].

2. Using `intpakX` to compute orbits of a chaotic dynamical system

In this section, we will demonstrate a simple but powerful application of `intpakX` to discrete dynamic systems (see also [9]).

The computation of an orbit of a *dynamic system* is known to be highly unstable if the system exhibits chaotic behavior. In this case, even for the very simplest systems, ordinary floating-point computations will eventually deliver results which are completely wrong quantitatively, when compared with the true trajectory on which the computation began. Similarly, ordinary interval arithmetic (i.e. intervals of floating-point numbers) yields poor enclosures after few iterations. In most cases the computation breaks down because of overflow. Using `intpakX`'s multiple precision

intervals, we can compute enclosures of orbits for a considerably longer time with high accuracy.

Consider the simple dynamic system as given by the *logistic equation*:

$$x_{n+1} = a \cdot x_n \cdot (1 - x_n)\,, \quad n \geq 0 \qquad (1)$$

for some $a \in [0, 4]$ and $x_0 \in (0, 1)$.

On the computer, we can compute this iteration with (i) ordinary floating-point arithmetic, (ii) ordinary interval arithmetic, or (iii) multiple precision interval arithmetic. However, for the cases (ii) and (iii) it would be better to first rewrite the right hand side of (1) such that it is better suited for the application of interval arithmetic: For narrow intervals it is well known in interval analysis that a tighter interval enclosure can be obtained by using a *mean value form* instead of an interval evaluation of the originally given expression.

The ordinary interval evaluation of a function $f(x)$ over an interval X, denoted as $f(X)$, is obtained via replacing all occurrences of x in f by the interval X and via replacing all operations by the corresponding interval operations. The mean value form is defined by $f_m(X) := f(y) + f'(X)(X - y)$ with some fixed value $y \in X$, e.g., the midpoint. Thus, in the cases (ii) and (iii) we may replace the right hand side of (1) by its mean value form, i.e. , by

$$X_{n+1} = a \cdot (y_n(1 - y_n) + (1 - 2X_n) \cdot (X_n - y_n)) \quad \text{with} \quad y_n \approx \text{mid}(X_n) = \text{midpoint of } X_n, \qquad (2)$$

where X_n is an interval in case (ii) and a multiple precision interval in case (iii). Rewriting (1) as (2) does not improve the quality of ordinary floating-point computation, which is still executed using (1).

The following Maple source code uses `intpakX` package to compute orbits for this equation. The results show the approximations x_n obtained by ordinary point evaluations of (1) and the enclosures X_n as obtained by multiple precision interval arithmetic using (1), and (2).

Repeat the computation of x_{500} using a 10, 15, 20, ... decimal digits arithmetic. The resulting approximations to x_{500} are all very different. Probably no one is close to the true (mathematical) value. The iteration exhibits chaotic behaviour. What is the true value of x_{500}?

```
>restart;
>for k from 10 by 5 to 90 do
>   Digits:= k;
>   nmax:= 500;
>   a:= 3.75;
>   x:= 0.5;
>   for n from 1 to nmax do
>       x:= a x (1 - x);
>   od;
>   Digits:= 10;
>   print(k, 1.0 x); #10 leading digits
>od;
              10, 0.9301133984
              15, 0.8270007028
              20, 0.8160192991
              25, 0.8788721414
              30, 0.8767493458
              35, 0.9316056390
              40, 0.4593190043
              45, 0.6394313121
              50, 0.5435332961
              55, 0.3102681501
              60, 0.7008367401
              65, 0.6438360607
              70, 0.2379718249
              75, 0.8715584686
              80, 0.3517828121
              85, 0.2767537630
              90, 0.2767538774
```

Simple idea: all iterates are rational numbers. Why not using Maple's rational number arithmetic? Let us try:

```
>nmax:= 15;
>     15
>a:= --;
>     4
>     1
>x:= -;
>     2
>for n from 1 to nmax do
>   st:= time();
>   x:= a x (1 - x);
>   nodd:= ceil(log10(op(1, x)))
>        + ceil(log10(op(2, x)));
>   if n < 5 then
>      print();
>      print(n, x, nodd, time() - st);
>   fi;
>   if n = 6 then print() fi;
>   if nmax - n < 5 then
>      print(n, nodd, time() - st) fi;
>od;
                        nmax := 15
                              15
                         a := --
                              4
                               1
                          x := -
                               2

                              15
                        1, --, 4, 0.008
                              16

                               225
                         2, ----, 7, 0.
                              1024

                             2696625
                       3, -------, 14, 0.008
                             4194304

                        60580179500625
                  4, --------------, 28, 0.004
                        70368744177664
```

```
11,  3698,  0.192
12,  7398,  0.748
13, 14796,  1.180
14, 29592,  3.853
15, 59184, 15.733
```

Numerators and denominators become very soon very large integers. To represent x_{15} 59184 decimal digits are necessary! Also, the computing time increases dramatically. It takes already 15 seconds to get x_{15}. This leads to the conclusion that using Maple it is not possible to compute more than say the first 20 iterates. It is far out of reach to compute (the rational number) x_{500}.

Now let us try to make use of the `intpakX` package. It provides a multiple precision interval arithmetic. Operator symbols denoting the interval operations are $\&+, \&-, \&*$, and $\&/$ (Maple does not allow operator overloading for the basic arithmetical operators).

```
>libname:=
>"/home/kraemer/intpakX/lib", libname
libname := "/home/kraemer/intpakX/lib",
  "/home/kraemer/maple10/lib"
>with(intpakX):       #use the package
>nmax:= 500;          #enclose x_500
>Digits:= nmax;       #use 500 digits
>a:= construct(3.75); #degenerate interval
>x:= construct(0.5);  #degenerate interval
>st:= time();         #start timer
>for n from 1 to nmax do
>    x := (a &* x) &* 1 &- x; #interval
>od;                          #expression
>time() - st;         #time used
>Digits := 10;
>1.0 &* x; #print a ten-digit enclosure
               nmax := 500
              Digits := 500
                   a := [3.75, 3.75]
  x := [0.5, 0.5]
                   0.356
      [0.2767538773, 0.2767538775]
```

We see a very satisfactory result. Using a 500 digits multiple precision interval arithmetic `intpakX` shows within one third of a second that $x_{500} \in [0.2767538773, 0.2767538775]$. Probably less digits would be enough (try to find the minimal number!).

As already pointed out: Using a Mean-Value-Form for the iteration function reduces the number of digits needed to get an accurate enclosure pretty much:

```
#Create Mean-Value-Form
> f:= (x,y)-> a*(y*(1-y)+(1-2*x)*(x-y));
#Create corresponding interval expr.
>F:= inapply(f(x), x):
f := proc (x, y)
   options operator, arrow;
   a*(y*(1-y)+(1-2*x)*(x-y))
end proc
>nmax:= 500;          #comupte x_500
>Digits:= 90;
>a:= construct(3.75); #degenerat interval
>x:= construct(0.5);
>for n from 1 to nmax do
>   y:= midpoint(x);
>   x:= F(x,y);       #use Mean-Value-Form
>od;
>Digits:= 10;
>1.0 &* x; #ten-digit enclosure
         nmax := 500
       Digits := 90
            a := [3.75, 3.75]
            x := [.5, .5]
  [.2767538753, .2767538798]
```

Using the Mean-Value-Form an 80 digits interval arithmetic leads to the enclosure $x_{500} \in [.2767538753, .2767538798]$.

Multiple precision interval computations allow to verify (long) orbits of the dynamical system whereas results computed by point computations may be totally incorrect.

3. Compute and visualize solution sets of systems of linear interval equations

In this section we denote by $I\mathbb{R}, I\mathbb{R}^n$, and $I\mathbb{R}^{m\times n}$ the sets of real intervals, of interval vectors with n components, and interval matrices with m rows and n columns, respectively.
To solve the interval linear system of equations

$$Ax = b\,,$$

with $A = [\underline{A}, \overline{A}] \in I\mathbb{R}^{n\times n}$ being an interval matrix and $b = [\underline{b}, \overline{b}] \in I\mathbb{R}^n$ being an interval vector, means to compute the solution set

$$\Sigma(A,b) := \{x \in \mathbb{R}^n \,|\, \dot{A}x = \dot{b} \text{ for some real } \dot{A} \in A, \dot{b} \in b\}. \quad (3)$$

For regular interval matrices A it holds

$$\Sigma(A,b) = \{\dot{A}^{-1}\dot{b} \in \mathbb{R}^n \,|\, \dot{A} \in A, \dot{b} \in b\}.$$

If A is not regular, the solution set may be empty or unbounded.

The computation of $\Sigma(A,b)$ is very costly. Therefore, in the field of selfverifying numerical methods often an interval vector including the true solution set is computed. The best possible one is the interval hull $\text{ihull}(\Sigma) := \bigcap_{X\in I\mathbb{R}^n, X\supseteq\Sigma} X$. But computing $\text{ihull}(\Sigma)$ is computationally still very costly [10]. In an interval setting the typical task is to compute a more or less sharp interval vector $z \in I\mathbb{R}^n$ containing $\text{ihull}(\Sigma)$.

Let us now discuss the Alefeld/Mayer/Rohn [14, 1] characterization of the solution set of an interval system of linear equations. Let O_k denote a closed orthant of $\mathbb{R}^n$ (there are 2^n orthants).

Theorem 1 *(Alefeld/Mayer [1], 1995) For regular interval matrices A it holds*

- $\Sigma(A,b)$ *is not convex.*
- *If* $\Sigma(A,b) \cap O_k \neq \emptyset$*, it is convex, compact, connected, and a polytope.*

3.1 How to compute $\Sigma(A,b) \cap O_k$?

A fixed orthant O is characterized by the sign vector $s = (s_i) \in S^n$ (i.e. $s_i \in \{-1,+1\}$ for $i = 1(1)n$) corresponding to the signs of the components of an interior point of O. Hence, if O denotes some orthant O, fixed by the signs $s_1, s_2, \ldots, s_n$, then $x = (x_i) \in O$ fulfills

$$x_i \begin{cases} \geq 0 \text{ if } s_i = +1, \\ \leq 0 \text{ if } s_i = -1. \end{cases}$$

For $i,j = 1(1)n$, let

$$c_{ij} := \begin{cases} \underline{a}_{ij} \text{ if } s_j = +1, \\ \overline{a}_{ij} \text{ if } s_j = -1 \end{cases}, d_{ij} := \begin{cases} \overline{a}_{ij} \text{ if } s_j = +1, \\ \underline{a}_{ij} \text{ if } s_j = -1. \end{cases}$$

Denote by $\underline{H}_i, \overline{H}_i, i = 1(1)n$, the half spaces

$$\underline{H}_i := \left\{ y \in I\!R^n | \sum_{j=1}^{n} c_{ij} y_j \leq \overline{b}_i \right\},$$

$$\overline{H}_i := \left\{ y \in I\!R^n | \sum_{j=1}^{n} d_{ij} y_j \geq \underline{b}_i \right\}.$$

Theorem 2 *(Rohn, 1989)*

$$\Sigma(A,b) \cap O = \bigcap_{i=1}^{n} (\underline{H}_i \cap \overline{H}_i) \cap O.$$

Note that $\underline{H}_i, \overline{H}_i$ depend on the choice of the orthant O (2^n possibilities). Denote the half spaces $\underline{H}_i, \overline{H}_i$ corresponding to orthant O_k by $\underline{H}_i^k, \overline{H}_i^k, k = 1(1)2^n$. Then we have the following representation of the exact solution set of the interval system of linear equations:

$$\Sigma(A,b) = \bigcup_{k=1}^{2^n} \underbrace{(\bigcap_{i=1}^{n} (\underline{H}_i^k \cap \overline{H}_i^k) \cap O_k)}_{\emptyset \text{ or a convex polytope}}.$$

To find the corners of the sets $\Sigma(A,b) \cap O_k$, the linear inequalities are transformed into systems of linear equations augmented by n further linear equations describing the actual orthant O_k. So for each orthant we get $\binom{3n}{n}$ equations. To find all corners of $\Sigma(A,b)$ we have to solve $2^n \binom{3n}{n}$ linear $n \times n$ point systems. Thus, for $n = 2$ we have to solve $60 \quad 2 \times 2$ and for $n = 3$ we have to solve $672 \quad 3 \times 3$ systems of linear equations.

Theorem 3 *The solution x to a (regular) point system is a corner of the convex set $\Sigma(A,b) \cap O_k$ if and only if x simultaneously fulfills all inequalities corresponding to the orthant O_k.*

3.2 How to plot $\Sigma(A,b)$?

We are now ready to outline the method used in the packages realized by Braun and Paw to compute the exact solution set of a linear inteval system in 2 or 3 dimensions:

a) Check whether $A \in I\!I\!R^n$ is regular:
   - $0 \notin \det A$ or
   - $\rho(|I - \dot{R}A|) < 1 \quad (\dot{R} \approx (\text{mid}(A))^{-1})$ (see [15])

   Give a warning if this check fails (note that it is still possible that the interval matrix of the system is regular).

b) Compute all vertices of the sets $\Sigma(A,b) \cap O_k$. This results in unordered sets of points in $I\!R^2$ or $I\!R^3$, respectively.

c) To plot the polytope in each orthant the vertices of each face of the polytope have to be found and sorted appropriately.

The spectral radius check $\rho(|I - \dot{R}A|) < 1$ must fail if the matrix A is not strongly regular [11]. As pointed out by a referee, checking regularity should be improved by a direct application of the Oettli/Prager Theorem [12]. So far, this check is not included in our package. For details of the implementation see the source codes of the packages [3, 13], which are available from the author.

3.3 Examples

The new command realized in the Braun package and used in this section is `IESolutionSet_With_Plots()`. Here, `IE` denotes interval equation. Parameters are the interval matrix, the interval right hand side, an optional list of orthants where the solution set is to be plotted, and an optional parameter `ShowPlanes`, additionally enabling the plotting of the coordinate planes within the same figure.

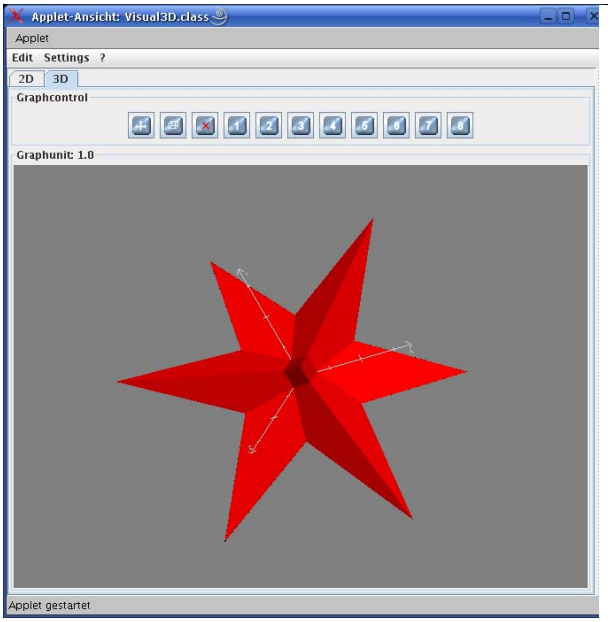


Figure 1. Neumaier's example.

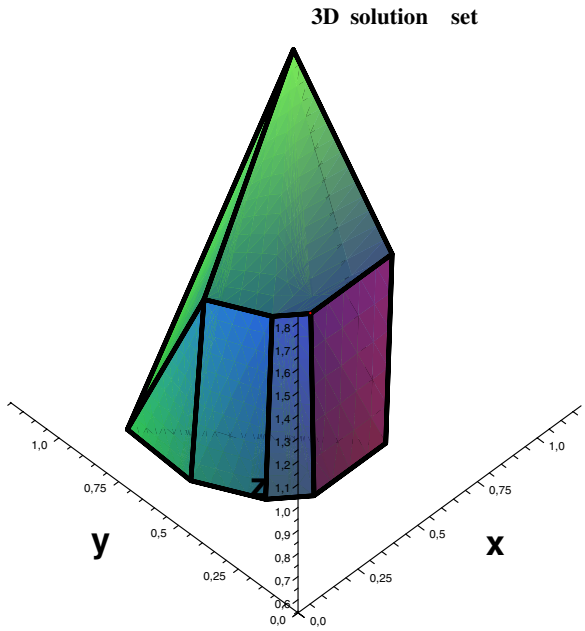


Figure 2. Most general shape of the solution set of a linear interval system restricted to a single orthant.

Figure 1 shows the 3D solution set of $Ax = b$ with

$$A := \begin{pmatrix} 3.5 & [0,2] & [0,2] \\ [0,2] & 3.5 & [0,2] \\ [0,2] & [0,2] & 3.5 \end{pmatrix}, \quad b := \begin{pmatrix} [-1,-1] \\ [-1,-1] \\ [-1,-1] \end{pmatrix}. \tag{4}$$

This example is taken from the book of Neumaier [11]. A corresponding graphics is shown on the cover sheet of the book cited. The figure shows a screen shot of the outcome of [13].

Parts of the 3D solution set of $Ax = b$ with

$$A := \begin{pmatrix} [-4,-3] & [1.5,2] & [0.1,0.1] \\ [-1,6] & [3,3] & [0.1,0.1] \\ [0,0] & [0.5,0.5] & [1,1] \end{pmatrix}, \quad b := \begin{pmatrix} [-1,0.75] \\ [0.5,2.5] \\ [1,1.8] \end{pmatrix} \tag{5}$$

are shown in Figure 2. Here, the intersection of the solution set with Orthant 1 is bounded by segments of 9 planes. Thus, Figure 5 shows the most general shape of such an intersection.

Figure 3 shows the 3D solution set of $Ax = b$ with

$$A := \begin{pmatrix} [-4,-3] & [1.5,2] & [0.1,0.2] \\ [-1,6] & [3,3] & [0,0] \\ [0,0] & [0.7,0.7] & [1,1] \end{pmatrix}, \quad b := \begin{pmatrix} [-1,0.75] \\ [0.5,2.5] \\ [1,1.8] \end{pmatrix}. \tag{6}$$

Again, the intersection of the solution set with the first orthant is bounded by segments of 9 planes.

Now we do no longer restrict the solution set to the first orthant. The full solution set of $Ax = b$ with

$$A := \begin{pmatrix} [-4,-3] & [1.5,2] & [0.1,0.1] \\ [-1,6] & [3,3] & [0,0] \\ [0,0] & [0.5,0.5] & [1,1] \end{pmatrix}, \quad b := \begin{pmatrix} [-1,0.75] \\ [0.5,2.5] \\ [1,1.8] \end{pmatrix}. \tag{7}$$

is shown in Figure 4.

The solution set corresponding to

$$A := \begin{pmatrix} 3 & [0,2] & 2 \\ [0,2] & 3 & [0,2] \\ 2 & [0,2] & 3 \end{pmatrix}, \quad b := \begin{pmatrix} [1,1] \\ [1,1] \\ [1,1] \end{pmatrix} \tag{8}$$

is shown in Figure 5.

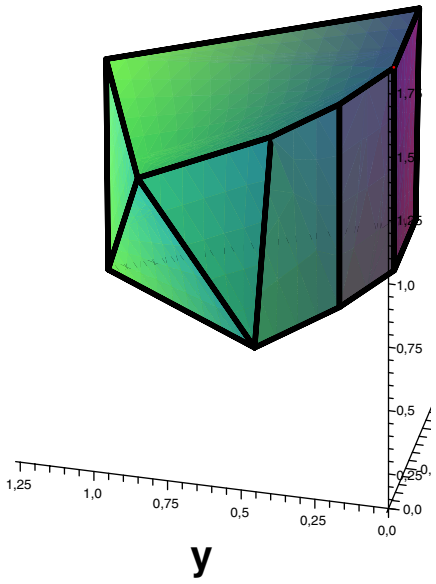


Figure 3. Most general shape in O thant 1.

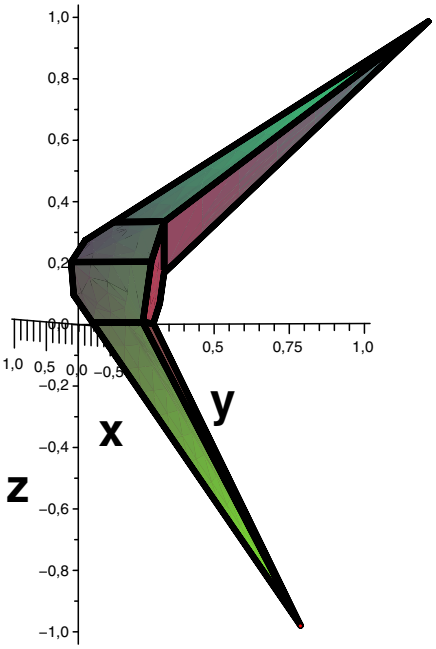


Figure 5. Complete 3D solution set corresponding to (8).

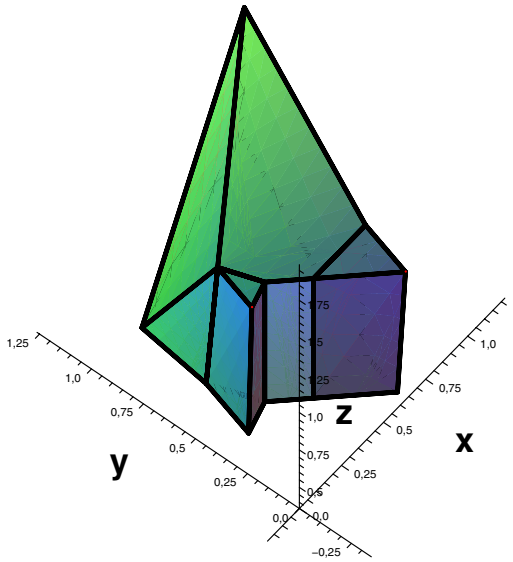


Figure 4. Complete 3D solution set.

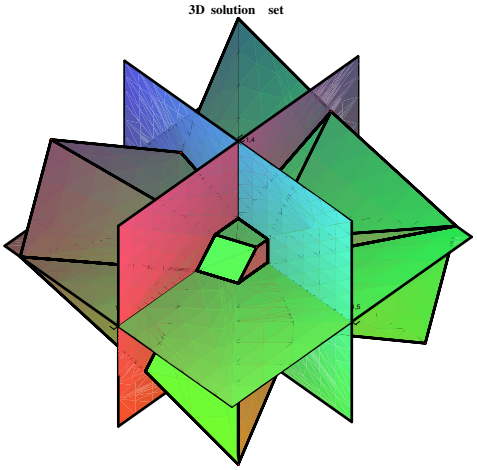


Figure 6. Complete solution set together with coordinate planes.

Using the (optional) parameter `ShowPlanes` when calling the new function `IESolutionSet_With_Plots()` allows to show not only the solution set of the linear interval system but also the coordinate planes (establihed by two different coordinate axes per plane). Figure 6 presents such a view for the linear interval sytsem (6). Note: The package is mainly intended for interactive use. The coloured visualization on the computer sreen and the additional possibilities to rotate, to zoom, to modify the transparency give a much better impression of the solution sets than the figures presented in this paper.

4. Conclusions

Bringing together capabilities of a computer algebra environment and well known methods from the field of reliable computing makes the `intpakX` package an outstanding scientific tool. Its source code (Maple programming language) is freely available. The package has been used by the author of this paper for several times in teaching selverifying numerical methods. The visualization facilities coming with `intpakX` are e.g. very well suited to investigate overestimations in range enclosures, verified root finding algorithms, solution sets of linear interval systems, and others. Of course, the package may be (and should be) extended in many directions. However, the dynamical system application presented in this paper shows that the implementation as it is can be used to solve non-trivial problems with mathematical rigor. Thus, `intpakX` is not only a valuable didactical tool but also a first step in showing that formula manipulation capabilities and selfverifying numerical methods should be combined in future releases of computer algebra packages like Maple, Mathematica, MuPad, and so on.

5. Acknowledgement

The author would like to thank his students/members of research group Ilse Geulig, Markus Grimmer, and Sven Braun for spending a fairly long time to complete the process of developing the package `intpakX`. Thanks also to Marc Janz, and Gregor Paw. Furthermore, the referees have been very helpful in improving the presentation of the paper.

References

[1] G. Alefeld and G. Mayer. On the symmetric and unsymmetric solution set of interval systems. *SIAM J. Matrix Anal. Appl.*, 16, 1995.

[2] W. Barth and E. Nuding. Optimale Lösung von Intervallgleichungssystemen. *Computing*, 12, 1974.

[3] S. Braun. Visualisierung der exakten Lösungsmengen von linearen Intervallgleichungssystemen in Maple. Master's thesis, University of Wuppertal, 2006.

[4] I. Geulig and W. Krämer. Intervallrechnung in Maple - Die Erweiterung `intpakX` zum Paket `intpak` der Share-Library. Technical report, Preprint IWRMM 1999/2, University of Karlsruhe, 1999.

[5] M. Grimmer. Interval arithmetic in Maple with `intpakX`. *PAMM - Proceedings in Applied Mathematics and Mechanics*, 2:442–443, January 2003.

[6] M. Kofler. *Maple: An Introduction and Reference*. Addison-Wesley, 1997.

[7] W. Krämer. Introduction to the Maple power tool `intpakX`. Technical report, Preprint BUW WRSWT 2006/9, University of Wuppertal, 2006. pp. 1–31. Accepted for publication in Bulgarian *Serdica Journal of Computing*, 2007.

[8] W. Krämer. Pitfalls in Maple. Technical report, Preprint BUW WRSWT 2006/7, University of Wuppertal, 2006.

[9] W. Krämer, R. Lohner, and U. Kulisch. *Numerical Toolbox for Verified Computing II – Advanced Numerical Problems*. Springer Verlag, to appear. Draft version, about 400 pages, 1995. Available from `lohner@rz.uni-karlsruhe.de`.

[10] V. Kreinovich and A. Lakeyev. Linear interval equations: Computing enclosures with bounded relative or absolute overestimation is np-hard. *Reliable Computing*, 2, 1996.

[11] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[12] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math.*, pages 405–409, 1964.

[13] G. Paw. Ein intuitiv bedienbares Java-Applet zur Visualisierung exakter Lösungsmengen von mengenwertigen numerischen Problemen. Master's thesis, University of Wuppertal, 2006.

[14] J. Rohn. Systems of linear interval equations. *Linear Algebra and Its Application*, 126:39–78, 1989.

[15] S. Rump. Verification methods for dense and sparse systems of equations. In *Topics in Validated Computations*. Elsevier, Amsterdam, 1994.

[16] . WRSWT research group. `intpakX`. Link at the University of Wuppertal: `http://www.math.uni-wuppertal.de/~xsc/software/intpakX/`.

Towards Interval Techniques for Processing Educational Data

Olga Kosheleva[1,2], Vladik Kreinovich[1,3], Luc Longpré[1,3],
Mourat Tschoshanov[1,2,4], and Gang Xiang[1,3]

[1]NASA Pan-American Center for Earth
and Environmental Studies
[2]Department of Teacher Education
[3]Department of Computer Science
[4]Department of Mathematical Sciences
University of Texas at El Paso
El Paso TX 79968, USA
contact olgak@utep.edu, vladik@utep.edu

Abstract

There are many papers that experimentally compare effectiveness of different teaching techniques. Most of these papers use traditional statistical approach to process the experimental results. The traditional statistical approach is well suited to numerical data but often, what we are processing is intervals (e.g., A means anything from 90 to 100). We show that the use of interval techniques leads to more adequate processing of educational data.

1. Formulation of the Problem

Practical problem: comparing teaching techniques. Teaching is very important, and teaching is not always very effective. There exist many different pedagogical techniques that help teach better, and new teaching techniques are being invented all the time.

To select the techniques which are the most efficient for a given educational environment, we must experimentally compare effectiveness of different teaching techniques in this and similar environments.

Setting up such an experiment in a meaningful way is a very difficult task. One needs to make sure that the students assigned to two different methods represent the same population, that the topics selected for the course are the same for both methods – otherwise, we will not get a fair and convincing comparison. In statistics, and especially in applications of statistics to social phenomena like teaching, there is a vast literature on experiment design.

In this paper, we will assume that the experiment has already been designed in a proper way, so that the comparison is reasonably fair, and we will concentrate on the next problem: how can we process the results of this experiment? based on these results, what conclusions can we make about the compared techniques?

A natural way to compare teaching techniques: compare the grades. A natural way to measure the effectiveness of a technique on an individual student is by recording the grade that this student received when being taught by this particular technique. To describe which method is better in general, it is therefore reasonable to describe the distribution of the grades for students taught under different techniques.

To get a complete picture, it is desirable to know the full grade distributions; however, in practice, researchers usually compute the mean and standard deviation.

In some cases, we have meaningful system of numerical grades. In some situations, there is a well-developed standardized test which provides a reasonably objective numerical measure of the student knowledge. For example, in the USA, there is the SAT test which is used to gauge the student's degree of preparation for undergraduate studies, the GRE test which is used to gauge the student's preparedness for graduate studies, the TOEFL test which gauges the degree to which non-native speakers know English, etc. Such tests provide a numerical grade from, say 0 to 800, and the difference between 640 and 650 is indeed meaningful.

The development of such a standardized test is a very difficult task. Moreover, once the test is given, and its answers are widely known, it is not possible to re-use it; so,

next time this test needs to be given, a new version of this test has to be designed. As a result, situations in which such standardized tests exist are extremely rare.

In most pedagogical situations, there is only a small number of possible grades. Typically, instructors assign grades from a small discrete set of possible grades. For example, in the US system, typical grades are A (excellent), B (good), C (satisfactory), D (sometimes used as passing grade), and F (fail); they are called *letter grades*.

In another system with which several of us are very familiar, namely, the Russian system, typical grades are 5 (excellent), 4 (good), 3 (satisfactory), 2 (bad), and 1 (sometimes used for awful). These grades are not described by letters; however, they serve the exact same purpose as the US letter grades. So, to enhance the useful distinction between meaningful numerical grades (like SAT scores) and grades from a small set, we will follow the US tradition and call grades from a small set *letter grades*.

Both in the US and in the Russian grading systems, there are sometimes refined versions of these systems with the possibility of adding + or $-$: e.g., 4+ is somewhat better than 4, A$-$ is somewhat worse than A.

Letter grades often comes from points. One of the main methods of assigning these letter grades is based on so-called *points*. Specifically, an instructor assigns points for different assignments and tests. These points add up during the course. At the end, the total amount of points determines the letter grade. There are many ways to translate from points to resulting grades. In the US, a typical translation is as follows:

- 90 points or above means A;
- at least 80 but less than 90 means B;
- at least 70 but less than 80 means C;
- at least 60 but less than 70 means D;
- less than 60 means F.

Why not keep the original points? The amount of points gained provides a much more refined description of the student knowledge than the resulting letter grade. So, a natural question is: why not simply use the original number of points instead of the letter grades?

The answer to this question is actually contained in the above explanation of why we cannot use too refined a scale: the main reason is the need to make the grades more objective. When several instructors teach different sections of a class, they spend a lot of effort trying to make sure that they have the same criteria for A, B, and C. When a department is accredited, accreditors looks at samples of A, B, and C papers to make sure that these sample would indeed get the corresponding letter grade at other schools as well.

This unification of letter grades is very difficult, so difficult that beyond letter grades, there is definitely no uniformity. A student who get a solid A (say, 95) with one of the instructors will most probably still get an A with others, but that A may vary from 91 to 99, depending on the instructor.

Because of this well-understood subjectivity, the points are usually not archived and not used to compare students taught by different instructors. Only letter grades are recorded.

Alternative approach: grades based on perceptions. In some disciplines, it is easy to reasonably objectively assign points to individual assignments: e.g., a well-formulated mathematical problem is either solved or not, its solution is either correct or not, and if there is a partial solution, it is possible to come up with an agreed-upon objective scale of points.

In other disciplines, however, e.g., in writing essays about a difficult philosophical concept, it is difficult to assign points. In such disciplines, it makes sense to make judgements like "understands well", "understands the main concept reasonably well, but still has misconceptions", etc. These judgments are then translated into points or directly into letter grades.

Again, instructors try their best to make these judgments objective. This unification is very difficult, so difficult that there is no way to extend this unification beyond the small scale of letter grades, into something more refined.

Statistical processing of letter grades: a problem. Let us get back to our problem. We want to compare two (or more) teaching techniques. So, we apply two different techniques to two groups of students, and we compare the results.

Because of the above explanations, at the end of the experiment, we only have letter grades: we have letter grades of students who were taught according to the first technique, and we have letter grades of students taught according to the second technique. To compare the techniques, we must perform a statistical analysis of these letter grades.

Statistical processing of letter grades: traditional approach. The traditional statistical approach to statistical processing of the grades is motivated by the existing ways of processing these data.

For example, students who study well receive different honors. These honors are usually based on simply averaging the grades: the higher the average, the higher the honors. In the Russian system, "letter grades" are actually numbers so they can be directly averaged. In the US system, there is a standard translation of letter grades into numbers: A means 4, B means 3, C means 2, D means 1, and F means 0. After this translation, letter grades become numbers, so we can easily compute their average.

Similarly, to gauge the degree to which a class learns a material, we can compute the average grade of this class.

In accordance with this idea, the vast majority of these papers that experimentally compare different teaching techniques by treating them as numbers:

- first, we translate the grades into numbers; and
- then, we process these numbers by using traditional statistical techniques; see, e.g., [18, 21].

Problems with the traditional approach: general description. The traditional statistical approach is well suited for processing numerical data. However, in processing educational data, often what we are processing is:

- either *intervals*: e.g.
  - the A grade usually means anything from 90 to 100,
  - the B grade means anything between 80 and 90, and
  - the C grade mean anything between 70 and 80;
- or *fuzzy*-type perceptions, words from the natural language like "understood well" or "understood reasonably well".

Problems with the traditional approach: example. In selecting a teaching method, it is important not only to make sure that the *average* results m are good – e.g., that the average grade on a standard test is good – but also to ensure that the results are *consistently* good – i.e., in statistical terms, that the standard deviation σ of the grade is low.

If the standard deviation σ is high, that would mean while some students learn really well under this technique, there are many others who are left behind, and we cannot afford that.

So, to compare several teaching techniques based on the grades the student got, we must compare not only their averages, but also the standard deviations.

The following simple example will show that when we replace an interval with a single value, we lose important information that could influence the computation of the standard deviation, and we could get erroneous results.

Suppose that in one method, all the students got Bs, while in the other method, half of the students got Bs and half of the students got As. Which of the two methods shows more stable results, with a smaller standard deviation?

In the traditional statistical approach, we interpret A as 4 and B as 3.

- In the first method, the resulting grades are $x_1 = \ldots = x_n = 3$, so the average grade is equal to

  $$m = \frac{x_1 + \ldots + x_n}{n} = 3,$$

  the population variance is equal to

  $$V = \sigma^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - m)^2 = 0,$$

  and the standard deviation is equal to $\sigma = \sqrt{V} = 0$;
- In the second method, the average is equal to

  $$m = \frac{3 + 4}{2} = 3.5,$$

  so for each i, $(x_i - m)^2 = 0.25$, hence the population variance is equal to

  $$V = \sigma^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - m)^2 = 0.25,$$

  and the standard deviation is equal to $\sigma = \sqrt{V} = 0.5$.

So, if we use the traditional statistical approach, we conclude that while the second method has a higher average, it is less stable than the first one.

In reality, if we go back from the "interval" letter grades like A, B, and C to the original numbers of points, it may turn out the second method is not only better on average, but also much more stable. Indeed, suppose that:

- in the first method, half of the students got 80 points, and half got 88 points; and
- in the second method, half of the students got 89 points, and half of the student got 91 points.

In terms of As and Bs, this is exactly the situation as described above. However, when we compute the standard deviation for these numbers of points, we get a different result than when we process the letter grades:

- In the first method, the average is equal to

  $$m = \frac{80 + 88}{2} = 84,$$

  so for each i, $(x_i - m)^2 = 16$, hence the variance is equal to

  $$V = \sigma^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - m)^2 = 16.$$

- In the second method, the average is equal to

$$m = \frac{89 + 91}{2} = 90,$$

so for each i, $(x_i - m)^2 = 1$, hence the variance is equal to

$$V = \sigma^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - m)^2 = 1 \ll 16.$$

What needs to be done. It is desirable to develop techniques for processing educational data that would take into account that the grades are not exactly equal to the corresponding numerical values but may differ from these values.

In other words, we need techniques that would provide guaranteed answers to questions like: Is the first method better than the second one?

It is OK to have an answer "we do not know", but if the answer is "yes", we want to be sure that no matter what additional information we learn about these experiments the answer will remain the same.

Such techniques are outlined in this paper.

2. Interval Approach

Processing interval data: analysis of the situation. The main reason why we had the above problem is that letter grade ℓ represents not a *single* value of the number grade x, but rather an *interval* $\mathbf{x} = [\underline{x}, \overline{x}]$ of possible values of the numbers of points. For example:

- the letter grade A represents the interval $[90, 100]$;
- the letter grade B represents the interval $[80, 90]$;
- the letter grade C represents the interval $[70, 80]$.

So, for the educational data, instead of the exact value x of each number of points, we often only know the intervals $[\underline{x}, \overline{x}]$ corresponding to the letter grade ℓ. This is true for the American system, this is true for the Russian system, this is true for any grading system in which the number of points is used to describe the resulting "letter grade".

Processing interval data: formulation of the problem. Our objective is, given a set of letter grades $\ell_1, \ldots, \ell_n$, to compute a certain statistical characteristic C such as average, standard deviation, correlation with other characteristics (such as the family income or the amount of time that a student spends on homeworks), etc.

The desired statistical characteristic is defined in terms of numerical values, as $C = C(x_1, \ldots, x_n)$. For example, the average is defined as $m = \frac{x_1 + \ldots + x_n}{n}$, the variance is defined as $V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - m)^2$, etc.

For the educational data, instead of the *exact* values x_i, we often only know the *intervals* $\mathbf{x}_i$ corresponding to the letter grade ℓ_i. For different possible values $x_1 \in \mathbf{x}_1$, …, $x_n \in \mathbf{x}_n$, we get different values of the corresponding characteristic C.

Our objective is to to find the range of possible values of the desired characteristic when $x_i \in \mathbf{x}_i$, i.e., the interval

$$\mathbf{C} = \{C(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

This problem is a particular case of the general problem of interval computations. The need to perform computations under interval uncertainty occurs in many areas of science and engineering. In many such areas, we therefore face the following problem:

- we know:
  - n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and
  - an algorithm $y = f(x_1, \ldots, x_n)$ that transforms n real numbers (inputs) into a single number y (result of data processing);
- we must estimate the range of possible values of y, i.e., the interval

$$\mathbf{y} = \{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

This problem is called the main problem of *interval computations*; see, e.g., [5, 6, 7, 10, 13].

We can therefore conclude that the problem of processing educational data under interval uncertainty is a particular case of the more general problem of interval computations.

How we can process interval data: general description. Many efficient techniques have been developed to solve generic interval computations problems; see, e.g., [5, 6, 7, 10, 13].

How we can process interval data: case of statistical characteristics. In particular, several algorithms have been developed for the case when the the function $f(x_1, \ldots, x_n)$ is one of the standard statistical characteristics such as average m or standard deviation V; see, e.g. [4, 11, 12] and references therein.

Computing average under interval uncertainty. In particular, since the average is a monotonic function of each of its variables, its value is the largest when each x_i attains the

largest possible value $x_i = \overline{x}_i$, and its value is the smallest when the variance attains its smallest possible value $\underline{x}_i$. Thus, for the average m, the interval takes the form $[\underline{m}, \overline{m}]$, where

$$\underline{m} = \frac{\underline{x}_1 + \ldots + \underline{x}_n}{n};\ \ \overline{m} = \frac{\overline{x}_1 + \ldots + \overline{x}_n}{n}.$$

If all the letter grades are A, B, C, or D, then the width $\overline{x}_i - \underline{x}_i$ of each corresponding interval is 10, so $\overline{m} = \underline{m} + 10$. In this situation, it is sufficient to compute *one* of the bounds $\overline{m}$ or $\underline{m}$, the other bound can be easily reconstructed from this one.

If one of the grades is a F grade, for which the interval of possible values is $[0, 60]$ with a width $60 > 10$, then we must compute *both* bounds.

Computing variance under interval uncertainty. For the variance V, there exist efficient algorithms for computing the lower bound $\underline{V}$, but the problem of computing the upper bound $\overline{V}$ is, in general, NP-hard. However, for educational data, the intervals either coincide or intersect at a single point; as a result, none of the intervals is a subset of the interior of any other: $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$. For the case when the intervals satisfy this *subset property*, there exists an efficient algorithms for computing $\overline{V}$; see, e.g., [12].

Specifically, to compute $\overline{V}$ for intervals which satisfy the subset property, we first sort the intervals in lexicographic order

$$[\underline{x}_i, \overline{x}_i] \preceq [\underline{x}_j, \overline{x}_j] \leftrightarrow \underline{x}_i < \underline{x}_j \vee (\underline{x}_i = \underline{x}_j \,\&\, \overline{x}_i \le \overline{x}_j).$$

For the points intervals, this simply means that we sort the letter grades into an increasing sequence. As a result, we get $\underline{x}_1 \le \underline{x}_2 \le \ldots \le \underline{x}_n$ and $\overline{x}_1 \le \overline{x}_2 \le \ldots \le \overline{x}_n$. For every k from 1 to n, we pick $x_i = \underline{x}_i$ for $i \le k$ and $x_i = \overline{x}_i$ for $i > k$; then, we compute the average $m = \frac{x_1 + \ldots + x_n}{n}$ of the selected x_i, and check whether this average satisfies the inequality $\underline{x}_k \le m \le \overline{x}_{k+1}$. If it does, then the population variance of the corresponding sequence $x_1, \ldots, x_n$ is exactly the desired upper bound $\overline{V}$.

According to [4, 12], to compute the lower bound $\underline{V}$, similarly, for every k, we select:

- $x_i = \overline{x}_i$ when $\overline{x}_i \le \underline{x}_k$, and
- $x_i = \underline{x}_i$ when $\underline{x}_i \ge \overline{x}_k$.

We then compute the average m of the selected x_i and check whether this average satisfies the inequality $\underline{x}_k \le m \le \overline{x}_k$. If it does, then we assign $x_i = m$ for all the un-assigned value i, and the population variance of the corresponding sequence $x_1, \ldots, x_n$ is exactly the desired lower bound $\underline{V}$.

Numerical example: computing $\overline{V}$. For 3 sorted grades C, B, and A, we get $\underline{x}_1 = 70$, $\overline{x}_1 = 80$, $\underline{x}_2 = 80$, $\overline{x}_2 = 90$, $\underline{x}_3 = 90$, $\overline{x}_3 = 100$. For this data, let us first compute $\overline{V}$. For $k = 1$, we pick $x_1 = \underline{x}_1 = 70$, $x_2 = \overline{x}_2 = 90$, and $x_3 = \overline{x}_3 = 100$. Here, $m = (x_1 + x_2 + x_3)/3 = 86\frac{2}{3}$. Since $\underline{x}_1 = 60 \le m \le \overline{x}_2 = 90$, the upper bound $\overline{V}$ is equal to the population variance $\frac{1}{n} \cdot \sum (x_i - m)^2$ of the values $x_1 = 70$, $x_2 = 90$, and $x_3 = 100$, hence $\overline{V} = 155\frac{5}{9}$.

Numerical example: computing $\underline{V}$. For $\underline{V}$, we also start with $k = 1$. For this k, in accordance with the above algorithm, we assign the values $x_2 = \underline{x}_2 = 80$ and $x_3 = \underline{x}_3 = 90$. Their average $m = 85$ is outside the interval $[\underline{x}_1, \overline{x}_1] = [70, 80]$, so we have to consider the next k.

For $k = 2$, we assign $x_1 = \overline{x}_1 = 80$ and $x_3 = \underline{x}_3 = 90$. The average $m = 85$ of these two values satisfies the inequality $\underline{x}_2 = 80 \le m \le \overline{x}_2 = 90$; hence we assign $x_2 = 85$, and compute $\underline{V}$ as the population variance of the values $x_1 = 80$, $x_2 = 85$, and $x_3 = 90$, hence $\underline{V} = 16\frac{2}{3}$.

Computing other statistical characteristics under interval uncertainty. Similar algorithms are known for other statistical characteristic such as median, higher moments, covariance, etc. [4, 11, 12].

3. Fuzzy Approach: In Brief

Formulation of the problem. In the interval approach, each letter grade is characterized by a set (interval) of possible values of points. For example, a letter grade A corresponds to the interval $[90, 100]$, etc.

On average, 90 is a reasonable and convenient threshold, but in reality, this threshold is somewhat arbitrary. Instructors often do not cut off at 90 sharp and assign A to all those who got 90.1 and B to those who got 89.9. In many cases, if there are good students whose grades are almost 90 (say, 89.1), they will get an A grade. In such cases, the threshold is made flexible: the instructor looks for a gap between clearly A and clearly B students (such a gap usually exists), and assigns A to all the students whose grades are higher than this gap and B to those students whose grades are below this gap.

As a result, even if a student has an A, we cannot say with 100% confidence that this student's number of points was above 90. The resulting situation can be described by the technique of fuzzy uncertainty see, e.g., [8, 16]. In this technique, for each number of points x and for each letter grade (e.g., A), we have a degree $\mu_A(x) \in [0, 1]$ with which x corresponds to A.

- When $x \ge 90$, we are absolutely sure that the letter grade is A, so $\mu_A(x) = 1$.

- When $x \le 87$, we are absolutely sure that the letter grade is not A, so $\mu_A(x) = 0$.
- When $87 < x < 90$, there is a possibility that A was assigned as a letter grade, so we get $\mu_A(x) \in (0, 1)$.

This value $\mu_A(x)$ is called a *membership degree* – the degree to which the value of x points is a member of the (fuzzy) set of all the values which correspond to the A grade.

To find these membership degrees, we can, e.g., use linear interpolation, and define $\mu_A(x)$ on the interval $[87, 90]$ as a linear function which takes the value 0 on $x = 87$ and the value 1 for $x = 90$.

For each membership α, we can determine the set of values x that are possible with at least this degree of possibility – the *α-cut* $\{x \mid \mu_A(x) \ge \alpha\}$ of the original fuzzy set. Vice versa, if we know α-cuts for every α, then, for each object x, we can determine the membership degree $\mu_A(x)$ with which x belongs to the original fuzzy set as the largest values α for which x beliongs to the corresponding α-cut [1, 8, 14, 15, 16].

A fuzzy set can be thus viewed as a nested family of its α-cuts.

How we can process fuzzy data: general idea. If instead of a (crisp) interval $\mathbf{x}_i$ of possible numbers of points, we have a fuzzy set $\mu_i(x)$ of possible grades, where i is A, B, C, D, or F, then we can view this information as a family of nested intervals $\mathbf{x}_i(\alpha)$ – α-cuts of the given fuzzy sets.

Our objective is then to compute the fuzzy number corresponding to this the desired characteristic $C(x_1, \ldots, x_n)$.

In this case, for each level α, to compute the α-cut of this fuzzy number, we can apply the interval algorithm to the α-cuts $\mathbf{x}_i(\alpha)$ of the corresponding fuzzy sets. The resulting nested intervals form the desired fuzzy set for C.

How we can process fuzzy data: case of statistical characteristics. For statistical characteristics such as variance, more efficient algorithms are described in [3].

4. Towards Combining Probabilistic, Interval, and Fuzzy Uncertainty

Need for such a combination. In the case of interval uncertainty, we consider all possible values of the grades, and do not make any assumptions about the probability of different values within the corresponding intervals. However, in many cases, we can make commonsense conclusions about the frequency of different grades.

For example, if a student has almost all As but only one B, this means that this is a strong student, and most probably this B is at the high end of the B interval. On the other hand, if a student has almost all Cs but only one B, this means that this is a weak student, and most probably this B is at the lower end of the B interval. It is desirable to take such arguments into account when processing educational data.

Let us describe how we can do this.

Comment. To avoid misunderstanding, it is worth mentioning that commonsense conclusions are not always possible: in some cases, we can make such commonsense conclusions; in some cases, we cannot. For example, if the first student has one A, two Bs and one C, and the second student has two As and two Cs, it is not clear which of two students is better, so it is difficult to make any conclusions about the quality of these grades.

Simplest case: normally distributed grades. Let us first consider the reasonable case when the actual points are normally distributed, with an (unknown) mean m and an unknown standard deviation σ. In other words, we assume that the cumulative probability distribution (cdf) $F(x) \stackrel{\text{def}}{=} \text{Prob}(\xi < x)$ has the form $F_0\left(\dfrac{x-m}{\sigma}\right)$, where $F_0(x)$ is the cdf of the standard Gaussian distribution with 0 mean and unit standard deviation. Our objective is to determine the values m and σ.

If we knew the values of the points x_i, then we could apply the above statistics and estimate m and $\sigma = \sqrt{V}$. In many situations, we do not know the values of the *points*, we only know the values of the *letter* grades. How can we then estimate m and σ based on these letter grades?

Case of normally distributed grades: towards an algorithm. Based on the letter grades, we can find, for the threshold values 60, 70, etc., the frequency with which we have the number of points smaller that this threshold. If we denote by f the proportion of F grades, by d the proportion of D grades, etc., then the frequency of $x < 60$ is f, the frequency of $x < 70$ is $f + d$, the frequency of $x < 60$ is $f + d + c$.

It is well known that the probability can be defined as a limit of the corresponding frequency when the sample size n increases. Thus, when the sample size is large enough, we can safely assume that the corresponding frequencies are close to the corresponding probabilities, i.e., to the values $F(x)$. In other words, we conclude that:

$$F_0\left(\frac{60-m}{\sigma}\right) \approx f;\ \ F_0\left(\frac{70-m}{\sigma}\right) \approx f+d;$$

$$F_0\left(\frac{80-m}{\sigma}\right) \approx f+d+c;\ \ F_0\left(\frac{90-m}{\sigma}\right) \approx f+d+c+b.$$

If we denote by $\psi_0(t)$ the function that is inverse to $F_0(t)$, then, e.g., the first equality takes the form $(60-m)/\sigma \approx \psi_0(f)$, i.e., $\sigma \cdot \psi_0(f) + m \approx 60$. Thus, to find the unknowns m and σ, we get a system of linear equations:

$$\sigma \cdot \psi_0(f) + m \approx 60;\ \ \sigma \cdot \psi_0(f+d) + m \approx 70;$$

$\sigma\cdot\psi_0(f+d+c)+m\approx 80;\ \ \sigma\cdot\psi_0(f+d+c+b)+m\approx 90,$

which can be solved, e.g., by using the Least Squares Method.

Comment. In some cases, the distribution is non-Gaussian, and we know its shape, i.e., we know that

$$F(x)=F_0\left(\frac{x-m}{\sigma}\right),$$

where $F_0(t)$ is a known function, and m and σ are unknown parameters. In this case, we can use the same formulas as above.

Simplified case when all the grades are C or above. In many cases, only C and above are acceptable letter grades. In such situations, $f=d=0$ and $c+b+a=1$, so we get a simplified system of two linear equations with two unknowns:

$$\sigma\cdot\psi_0(c)+m=80;\ \ \sigma\cdot\psi_0(c+b)+m=90.$$

Subtracting the first equation from the second one, we conclude that

$$\sigma=\frac{10}{\psi_0(b+c)-\psi_0(c)}.$$

This formula can be further simplified if the distribution $F_0(x)$ is symmetric (e.g., Gaussian distribution is symmetric), i.e., for every x, the probability $F_0(-x)$ that $\xi\le -x$ is equal to the probability $1-F_0(x)$ that $\xi\ge x$. Thus, we can conclude that $\psi_0(1-x)=-\psi_0(x)$ for every x. In particular, since $c+b+a=1$, we conclude that

$$-\psi_0(c+b)=\psi_0(1-(c+b))=\psi_0(a).$$

Thus, the formula for σ takes the form:

$$\sigma=-\frac{10}{\psi_0(a)+\psi_0(c)}. \tag{1}$$

Similarly, if we divide the equation $(90-m)/\sigma=\psi_0(b+c)$ by $(80-m)/\sigma=\psi_0(c)$, we conclude that

$$\frac{90-m}{80-m}=\frac{\psi_0(b+c)}{\psi_0(c)}=-\frac{\psi_0(a)}{\psi_0(c)},$$

hence

$$m=80+\frac{10}{1+\dfrac{\psi_0(a)}{\psi_0(c)}}. \tag{2}$$

Relation to fuzzy logic. As we can see from the formulas (1) and (2), the standard deviation is an increasing function of the sum $\psi_0(a)+\psi_0(c)$, while the mean m is monotonically increasing with the ratio $\psi_0(a)/|\psi_0(c)|$. This makes sense of we take into account that $\psi_0(a)$ monotonically depends on the proportion a of grades in the A range: the more grades are in the A range and the fewer grades are in the C range, the larger the average grade m, so m should be kind of monotonically depending on the degree to which it is true that we have A grades and not C grades.

It is worth mentioning that the operations of sum as "or" and ratio as "a and not c" appear when we try to interpret neural networks in terms of fuzzy logic [2]; see also Appendix.

5. Conclusions and Future Work

Processing incomplete pedagogical data: formulation of the problem and what we did (a brief summary). In the above text, we considered the following pedagogical situation:

- we have two (or more) techniques for teaching the same material;
- we have a way to gauge the degree to which students learned this material;
- we need to select a techniques for which, on average, the students learn better.

In this paper, we consider the situation in which the degree to which a student learned the material is determined by this student's letter grade for this class. This letter grade is, of course, an incomplete description of the student's knowledge. In this paper, we described new algorithms which take into account this incompleteness.

Natural next question: how can we make the pedagogical data more complete? From the mathematical and computational viewpoint, the above setting already leads to non-trivial computational problems. From this viewpoint, the natural next idea may be to improve these algorithms, make them more efficient and more general – in other words, how to best process the existing incomplete data.

From the pedagogical viewpoint, however, a natural next question is how we can *supplement* the letter grades to get a more complete picture of the students' knowledge.

How to make pedagogical data more complete: main idea. For most classes, a large part of the material studied in the class is important not (or not only) only by itself, but (also) because it provides a basis for the important things which will be learned in the following classes.

This fact was emphasized by Vygotsky (see, e.g., [20]), according to whom the class' success is determined not only by the students' mastery of the class material, but also by the increased student abilities to learn new related material. To enhance transition to next classes, it is desirable to prepare students for future courses by appropriate examples.

Example. Success in high school mathematics can be judged not only by the students' grades on the corresponding subjects, but also by how prepared the students are in the long run for studying advanced topics such as calculus. From this viewpoint, it is desirable to include simple optimization and area-computational exercises in algebra and geometry – as examples for which later-learned calculus techniques will work much faster [19].

Related future work. To test the success of this technique, to compare different techniques of teaching lower-level classes (e.g., algebra or geometry), we must take into account not only the student grades in these classes, but also the students' grades in the following more advanced classes (e.g., calculus).

In view of this need, we must extend our statistical techniques in such a way that they not only take into account interval, fuzzy, and probabilistic uncertainty in the letter grades for the current class, but also the corresponding uncertainty in letter grades for the future classes.

Acknowledgments. This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Star Award from the University of Texas System, Texas Department of Transportation grant No. 0-5453, and the University Research Institute grant from the University of Texas at El Paso.

The authors are thankful to participants of SCAN'06 for valuable discussions, and to the anonymous referees for important suggestions.

References

[1] G. Bojadziev and M. Bojadziev. *Fuzzy Sets, Fuzzy Logic, Applications*, World Scientific, Singapore, 1995.

[2] S. Dhompongsa, V. Kreinovich, and H. T. Nguyen. How to interpret neural networks in terms of fuzzy logic?, In: *Proceedings of the Second Vietnam-Japan Bilateral Symposium on Fuzzy Systems and Applications VJFUZZY'2001*, Hanoi, Vietnam, December 7–8, 2001, pp. 184–190.

[3] D. Dubois, H. Fargier, and J. Fortin J. The empirical variance of a set of fuzzy intervals, In: *Proceedings of the 2005 IEEE International Conference on Fuzzy Systems FUZZ-IEEE'2005*, Reno, Nevada, May 22–25, 2005, pp. 885–890.

[4] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles. Exact bounds on finite populations of interval data. *Reliable Computing*, 11(3):207–233, 2005.

[5] L. Jaulin, M. Keiffer, O. Didrit, and E. Walter. *Applied Interval Analysis*, Springer-Verlag, London, 2001.

[6] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.

[7] R. B. Kearfott and V. Kreinovich (eds.). *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

[8] G. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall, New Jersey, 1995.

[9] O. M. Kosheleva and M. Ceberio. Processing educational data: from traditional statistical techniques to an appropriate combination of probabilistic, interval, and fuzzy approaches. In: *Proceedings of the International Conference on Fuzzy Systems, Neural Networks, and Genetic Algorithms FNG'05*, Tijuana, Mexico, October 13–14, 2005, pp. 39–48.

[10] V. Kreinovich, D. Berleant, and M. Koshelev, website on interval computations
http://www.cs.utep.edu/interval-comp

[11] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos. Interval versions of statistical techniques, with applications to environmental analysis, bioinformatics, and privacy in statistical databases. *Journal of Computational and Applied Mathematics*, 199(2):418–423, 2007.

[12] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpré, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos. Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity. *Reliable Computing*, 12(6):471–501, 2006.

[13] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[14] R. E. Moore and W. A. Lodwick. Interval analysis and fuzzy set theory. *Fuzzy Sets and Systems*, 135(1):5–9, 2003.

[15] H. T. Nguyen and V. Kreinovich. Nested intervals and sets: concepts, relations to fuzzy sets, and applications. In: R. B. Kearfott and V. Kreinovich (eds.). *Applications of Interval Computations*, Kluwer, Dordrecht, 1996, pp. 245–290.

[16] H. T. Nguyen and E. A. Walker. *A First Course in Fuzzy Logic*. CRC Press, Boca Raton, Florida, 2005.

[17] M. Q. Patton. *Qualitative Research and Evaluation Methods*, Sage Publ., Thousand Oaks, California, 2002.

[18] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[19] M. Tchoshanov, S. Blake, and A. Duval. Preparing teachers for a new challenge: Teaching Calculus concepts in middle grades. In: *Proceedings of the Second International Conference on the Teaching of Mathematics (at the undergraduate level)*, Hersonissos, Crete, Greece, 2002.

[20] L. Vygotsky. *Thought and Language*. M.I.T. Press, Cambridge, Massachusetts, 1962.

[21] H. M. Wadsworth Jr (ed.) *Handbook of Statistical Methods for Engineers and Scientists*. McGraw-Hill Publishing Co., New York, 1990.

A. Statistical Formulas Have a Fuzzy Meaning: Detailed Explanation

Reminder. In the main text, we have derived statistical formulas for the mean and standard deviation, and we mentioned that these formulas can be interpreted in terms of fuzzy logic – if we use sum as "or" and ratio as "a and not c".

Comment. In principle, we could use arbitrary fuzzy t-norms, t-conorms, and fuzzy negations, but then we would

then not get the statistical formulas. Our point is not that we can use different fuzzy operations, but that *statistical* formulas can be interpreted in terms of reasonable *fuzzy* operations.

Selecting an "or" operation. The degree of belief a in a statement A can be estimated as proportional to the number of arguments in favor of A. In principle, there exist infinitely many potential arguments, so in general, it is hardly probable that when we pick a arguments out of infinitely many and then b out of infinitely many, the corresponding sets will have a common element. Thus, it is reasonable to assume that every argument in favor of A is different from every argument in favor of B. Under this assumption, the total number of arguments in favor of A and arguments in favor of B is equal to $a + b$. Hence, the natural degree of belief in $A \vee B$ is proportional to $a + b$.

Selecting an "and" operation. Different experts are reliable to different degrees. Our degree of belief in a statement A made by an expert is equal to $w \,\&\, a$, where w is our degree of belief in this expert, and a is the expert's degree of belief in the statement A. What are the natural properties of the "and"-operation?

First, since $A \,\&\, B$ means the same as $B \,\&\, A$, it is reasonable to require that the corresponding degrees $a \,\&\, b$ and $b \,\&\, a$ should coincide, i.e., that the "and"-operation be commutative.

Second, when an expert makes two statements B and C, then our resulting degree of belief in $B \vee C$ can be computed in two different ways:

- We can first compute *his* degree of belief $b \vee c$ in $B \vee C$, and then us the "and"-operation to generate our degree of belief $w \,\&\, (b \vee c)$.
- We can also first generate our degrees $w \,\&\, b$ and $w \,\&\, c$, and then use an "or"-operation to combine these degrees, arriving at $(w \,\&\, b) \vee (w \,\&\, c)$.

It is natural to require that both ways lead to the same degree of belief, i.e., that the "and"-operation be distributive with respect to $\vee$.

It is also reasonable to assume that the value $w \,\&\, a$ is a monotonically (non-strictly) increasing function of each its variables.

It can be shown [2] that every commutative, distributive, and monotonic operation $\& : R \times R \to R$ has the form $a \,\&\, b = C \cdot a \cdot b$ for some $C > 0$. This expression can be further simplified if we introduce a new scale of degrees of belief $a' \stackrel{\text{def}}{=} C \cdot a$; in the new scale, $a \,\&\, b = a \cdot b$.

Selecting a crisp truth value. We know that "true" and "true" is "true", and that "false" and "false" is "false". Thus, it is reasonable to call a positive degree of belief e_0 is a crisp value if $e_0 \,\&\, e_0 = e_0$.

This implies that $e_0 = 1$.

Selecting implication and negation. From the commonsense viewpoint, an implication $A \to B$ is a statement C such that if we add C to B, we get A. Thus, it is natural to define an *implication operation* as a function $\to : R \times R \to R$ for which, for all a and b, we have $(a \to b) \,\&\, a = b$. One can easily check that $a \to b = b/a$.

Negation $\neg A$ can be viewed as a particular case of implication, $A \to F$, for a crisp (specifically, false) value F. Thus, we can define negation operation as $a \to e_0$, i.e., as $1/a$.

Towards Combining Probabilistic, Interval, Fuzzy Uncertainty, and Constraints: An Example Using the Inverse Problem in Geophysics

V. Kreinovich[1], S. A. Starks[2], R. Araiza[1], and G. Xiang[1]
Depts. [1]Computer Science and [2]Electrical & Computer Eng.
Univ. of Texas at El Paso
El Paso, TX 79968, USA
contact vladik@utep.edu

A. A. Velasco and M. G. Averill
Dept. Geological Sci.
Univ. of Texas at El Paso
El Paso, TX 79968, USA
mgaverill@utep.edu

G. R. Keller
School of Geology & Geophysics
University of Oklahoma
100 East Boyd
Norman, OK 73019, USA
grkeller@ou.edu

Abstract

In many real-life situations, we have several types of uncertainty: measurement uncertainty can lead to probabilistic and/or interval uncertainty, expert estimates come with interval and/or fuzzy uncertainty, etc. In many situations, in addition to measurement uncertainty, we have prior knowledge coming from prior data processing and/or prior knowledge coming from prior interval constraints.

In this paper, on the example of the seismic inverse problem, we show how to combine these different types of uncertainty.

1. Seismic Inverse Problem: A Brief Description

In evaluations of natural resources and in the search for natural resources, it is very important to determine Earth structure. Our civilization greatly depends on the things we extract from the Earth, such as fossil fuels (oil, coal, natural gas), minerals, and water. Our need for these commodities is constantly growing, and because of this growth, they are being exhausted. Even under the best conservation policies, there is (and there will be) a constant need to find new sources of minerals, fuels, and water.

The only sure-proof way to guarantee that there are resources such as minerals at a certain location is to actually drill a borehole and analyze the materials extracted. However, exploration for natural resources using indirect means began in earnest during the first half of the 20th century. The result was the discovery of many large relatively easy to locate resources such as the oil in the Middle East.

However, nowadays, most easy-to-access mineral resources have already been discovered. For example, new oil fields are mainly discovered either at large depths, or under water, or in very remote areas – in short, in the areas where drilling is very expensive. It is therefore desirable to predict the presence of resources as accurately as possible before we invest in drilling.

From previous exploration experiences, we usually have a good idea of what type of structures are symptomatic for a particular region. For example, oil and gas tend to concentrate near the top of natural underground domal structures. So, to be able to distinguish between more promising and less promising locations, it is desirable to determine the structure of the Earth at these locations. To be more precise, we want to know the structure at different depths z at different locations (x, y).

Data that we can use to determine the Earth structure. In general, to determine the Earth structure, we can use different measurement results that can be obtained without actually drilling the boreholes: e.g., gravity and magnetic measurements, analyzing the travel-times and paths of seismic ways as they propagate through the earth, etc.

To get a better understanding of the Earth structure, we must rely on *active* seismic data – in other words, we must make artificial explosions, place sensors around them, and measure how the resulting seismic waves propagate. The most important information about the seismic wave is the *travel-time* t_i, i.e., the time that it takes for the wave to travel from its source to the sensor. To determine the geophysical structure of a region, we measure seismic travel times and reconstruct velocities at different depths from these data. The problem of reconstructing this structure is called the *seismic inverse problem*.

2. Known Algorithms for Solving the Seismic Inverse Problem: Description, Successes, Limitations

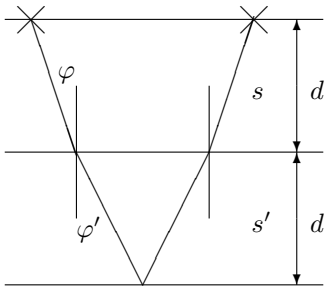
We want to find the values of the velocity $v(\vec{x})$ at different 3-D points $\vec{x}$. Based on the finite number of measurements, we can only reconstruct a finite number of parameters. So, we use a rectangular grid structure to divide the 3-D volume into box-shaped cells. We assume that the value of the velocity v_j is the same within each cell, and we reconstruct the velocities v_j within different cells.

Algorithm for the forward problem: brief description. Once we know the velocities v_j in each cell j, we can then determine the paths which seismic waves take. Seismic waves travel along the shortest path – shortest in terms of time. It can be easily determined that for such paths, within each cell, the path is a straight line, and on the border between the two cells with velocities v and v', the direction of the path changes in accordance with Snell's law $\frac{\sin(\varphi)}{v} = \frac{\sin(\varphi')}{v'}$, where φ and φ' are the angles between the paths and the line orthogonal to the border between the cells. (If this formula requires $\sin(\varphi') > 1$, this means that this wave cannot penetrate into the neighboring cell at all; instead, it bounces back into the original cell with the same angle φ.)

In particular, we can thus determine the paths from the source to each sensor. The travel-time t_i along i-th path can then be determined as the sum of travel-times in different cells j through which this path passes: $t_i = \sum\limits_j \frac{\ell_{ij}}{v_j}$, where ℓ_{ij} denotes the length of the part of i-th path within cell j.

This formula becomes closer to linear if we replace the original unknowns – velocities v_j – by their inverses $s_j \stackrel{\text{def}}{=} \frac{1}{v_j}$, called *slownesses*. In terms of slownesses, the formula for the travel-time takes the simpler form $t_i = \sum\limits_j \ell_{ij} \cdot s_j$.

It is worth mentioning, however, that the resulting system of equations is *not* linear in the unknowns s_j. Indeed, the actual geometry of the shortest path between the two given points depends on the actual values of the velocities v_j – i.e., equivalently, on the slownesses s_j. Thus, the lengths ℓ_{ij} of the segments of these shortest paths also depend on the slownesses $s_1, \ldots, s_m$. To be more precise, we should therefore explicitly take this dependence into account and re-write the above system as $t_i = \sum\limits_j \ell_{ij}(s_1, \ldots, s_m) \cdot s_j$ for an appropriate non-linear dependence $\ell_{ij}(s_1, \ldots, s_m)$.



Algorithm for the inverse problem: general description. There are several algorithms for solving this inverse problem; see, e.g., [11, 24, 28]. The most widely used is the following iterative algorithm proposed by John Hole [11].

At each stage of this algorithm, we have some approximation to the desired slownesses. We start with some reasonable initial slownesses, and we hope that after several iterations, we will be able to get slownesses which are much closer to the actual values.

At each iteration, we first use the currently known slownesses s_j to find the corresponding paths from the source to each sensor. Based on these paths, we compute the predicted values $t_i = \sum\limits_j \ell_{ij} \cdot s_j$ of travel-times.

Since the currently known slownesses s_j are only approximately correct, the travel-times t_i (which are predicted based on these slownesses) are approximately equal to the measured travel-times $\widetilde{t}_i$; there is, in general, a discrepancy $\Delta t_i \stackrel{\text{def}}{=} \widetilde{t}_i - t_i \neq 0$. It is therefore necessary to use these discrepancies to update the current values of slownesses, i.e., replace the current values s_j with corrected values $s_j + \Delta s_j$. The objective of this correction is to eliminate (or at least decrease) the discrepancies $\Delta t_i \neq 0$. In other words, the objective is to make sure that for the corrected values of the slowness, the predicted travel-times are closer to $\widetilde{t}_i$.

Of course, once we have changed the slownesses, the shortest paths will also change; however, if the current values of slownesses are reasonable, the differences in slowness are not large, and thus, paths will not change much. Thus, in the first approximation, we can assume that the paths are the same, i.e., that for each i and j, the length ℓ_{ij} remains the same. In this approximation, the new travel-times are equal to $\sum \ell_{ij} \cdot (s_j + \Delta s_j)$. The desired condition is then $\sum \ell_{ij} \cdot (s_j + \Delta s_j) = \widetilde{t}_i$. Subtracting the formula $t_i = \sum\limits_j \ell_{ij} \cdot s_j$ from this expression, we conclude that the corrections Δs_j must satisfy the following system of (approximate) linear equations: $\sum \ell_{ij} \cdot \Delta s_j \approx \Delta t_i$.

Solving this system of linear equations is not an easy task, because we have many observations and many cell values and thus, many unknowns, and for a system of linear equations, computation time required to solve it grows

as a cube c^3 of the number of variables c. So, instead of the standard methods for solving a system of linear equations, researchers use special faster geophysics-motivated techniques (described below) for solving the corresponding systems. These methods are described, in detail, in the next subsection.

Once we solve the corresponding system of linear equations, we compute the updated values Δs_j, compute the new (corrected) slownesses $s_j + \Delta s_j$, and repeat the procedure again. We stop when the discrepancies become small; usually, we stop when the mean square error $\frac{1}{n}\sum_{i=1}^{n}(\Delta t_i)^2$ no longer exceeds a given threshold. This threshold is normally set up to be equal to the measurement noise level, so that we stop iterations when the discrepancy between the model and the observations falls below the noise level – i.e., when, for all practical purposes, the model is adequate.

Algorithm for the inverse problem: details. Let us describe, in more detail, how the above auxiliary linear system of equations with unknown Δs_j is usually solved. In other words, for a given cell j, how do we find the correction Δs_j to the current value of slowness s_j in this cell?

Let us first consider the simplified case when there is only path, and this path is going through the j-th cell. In this case, cells through which this path does not go do not need any correction. To find the corrections Δs_j for all the cells j through which this path goes, we only have one equation $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$. The resulting system of linear equations is clearly under-determined: we have a single equation to find the values of several variables Δs_j. Since the system is under-determined, we have a infinite number of possible solutions. Our objective is to select the most geophysical reasonable of these solutions.

For that, we can use the following idea. Our single observation involves several cells; we cannot distinguish between the effects of slownesses in different cells, we only observe the overall effect. Therefore, there is no reason to assume that the value Δs_j in one of these cells is different from the values in other cells. It is thus reasonable to assume that all these values are close to each other: $\Delta s_j \approx \Delta s_{j'}$. The least squares method enables us to describe this assumption as minimization of the objective function $\sum\limits_{j,j'}(\Delta s_j - \Delta s_{j'})^2$ under the condition that $\sum \ell_{ij} \cdot \Delta s_j = \Delta t_i$. The minimum is attained when all the values Δs_j are equal. Substituting these equal values into the equation $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$, we conclude that $L_i \cdot \Delta s = \Delta t_i$, where $L_i = \sum\limits_j \ell_{ij}$ is the overall length of i-th path. Thus, in the simplified case in which there is only one path, to the slowness of each cell j along this path, we add the same value $\Delta s_j = \dfrac{\Delta t_i}{L_i}$.

Let us now consider the realistic case in which there are many paths, and moreover, for many cells j, there are many paths i which go through the corresponding cell. For a given cell j, based on each path i passing through this cell, we can estimate the correction Δs_j by the corresponding value $\Delta s_{ij} \stackrel{\text{def}}{=} \dfrac{\Delta t_i}{L_i}$. Since there are usually several paths going through the j-th cell, we have, in general, several different estimates $\Delta s_j \approx \Delta s_{ij}$. Again, the least squares approach leads to $\sum\limits_i (\Delta s_j - \Delta s_{ij})^2 \to \min$, hence to Δs_j as the arithmetic average of the values Δs_{ij}.

Comment. To take into account that paths with larger ℓ_{ij} provide more information, researchers also used weighted average, with weight increasing with ℓ_{ij}.

Successes of the known algorithms. The known algorithms have been actively used to reconstruct the slownesses, and, in many practical situations, they have led to reasonable geophysical models.

Limitations of the known algorithms. Often, the velocity model that is returned by the existing algorithm is not geophysically meaningful: e.g., it predicts velocities outside of the range of reasonable velocities at this depth. To avoid such situations, it is desirable to incorporate the expert knowledge into the algorithm for solving the inverse problem.

In our previous papers [2, 3, 13], we described how to do it. Specifically, we proposed a $O(c \log(c))$ time algorithm for taking interval prior knowledge into account.

In this paper, we provide a detailed motivation for that algorithm, and we use this motivation to design a new, faster, linear-time ($O(c)$) for solving this problem.

3. Case of Interval Prior Knowledge: Description and Known Algorithm

Interval prior knowledge. For each cell j, a geophysicist often provides us with his or her estimate of possible values of the corresponding slowness s_j. Often, this estimates comes in the form of an interval $[\underline{s}_j, \overline{s}_j]$ that is guaranteed to contain the (unknown) actual value of slowness.

It is desirable to modify Hole's algorithm in such a way that on all iterations, slownesses s_j stay within the corresponding intervals. Such a modification is described in [2, 3, 13].

Analysis of the problem and our main idea. Once we know the current approximations $s_j^{(k)}$ to slownesses, then,

along each path i, we want to find the corrections Δs_{ij} which provide the desired compensation, i.e., for which

$$\sum_{j=1}^{c} \ell_{ij} \cdot \Delta s_{ij} = \Delta t_i. \tag{1}$$

In Hole's algorithm, we select $\Delta s_{ij} = \dfrac{\Delta t_i}{L_i}$. With the additional knowledge, we may not be able to do this, because we want to make sure that the corrected values of slowness stay within the corresponding intervals

$$\underline{s}_j \le s_j^{(k)} + \Delta s_{ij} \le \overline{s}_j, \tag{2}$$

i.e., equivalently, that

$$\underline{\Delta}_j \le \Delta s_{ij} \le \overline{\Delta}_j, \tag{3}$$

where $\underline{\Delta}_j \stackrel{\text{def}}{=} \underline{s}_j - s_j^{(k)}$ and $\overline{\Delta}_j \stackrel{\text{def}}{=} \overline{s}_j - s_j^{(k)}$. Since $s_j^{(k)} \in [\underline{s}_j, \overline{s}_j]$, we conclude that $\underline{\Delta}_j \le 0$ and $\overline{\Delta}_j \ge 0$ – i.e., all lower endpoints are non-positive and all upper endpoints are non-negative.

How can we achieve this goal?

For each cell j, after an iteration of, say, Hole's algorithm, we have a corrected value of the slowness $s_j^{(k+1)} = s_j^{(k)} + \Delta s_{ij}$ which approximates the actual (unknown) slowness s_j: $s_j \approx s_j^{(k+1)}$. We also know that s_j should be located in the interval $[\underline{s}_j, \overline{s}_j]$. Similar to our previous analysis, it is therefore reasonable to use the Least Squares Method to combine these two piece of information: i.e., we look for the value $s_j \in [\underline{s}_j, \overline{s}_j]$ for which the square $(s_j - s_j^{(k)})^2$ is the smallest possible. In geometric terms, we look for the value within the given interval $[\underline{s}_j, \overline{s}_j]$ which is the closest to $s_j^{(k+1)}$. Thus:

- If the value $s_j^{(k+1)}$ is already within the interval, we keep it intact.
- If the value $s_j^{(k+1)}$ is to the left of the interval, i.e., if $s_j^{(k+1)} < \underline{s}_j$, then the closest point from the interval is its left endpoint $\underline{s}_j$.
- Similarly, if the value $s_j^{(k+1)}$ is to the right of the interval, i.e., if $s_j^{(k+1)} > \overline{s}_j$, then the closest point from the interval is its right endpoint $\overline{s}_j$.

In other words, e.g., for $\Delta t_i > 0$, we first find the universal value Δs and then, for those j for which $\Delta s > \overline{\Delta}_j$, we replace this value with $\overline{\Delta}_j$.

As a result, we arrive at the values Δs_{ij} which are all equal to Δs – except for those values for which $\overline{\Delta}_j < \Delta s$; for these values, $\Delta s_{ij} = \overline{\Delta}_j$.

Complications coming from a straightforward application of this idea. Originally, before we took interval prior knowledge into account, we had a full compensation for Δt_i. Now that we decreased some slownesses Δs_{ij}, the resulting value of $\sum\limits_{j=1}^{c} \ell_{ij} \cdot \Delta s_{ij}$ is, in general, smaller than Δt_i. Thus, there is a remaining discrepancy $\Delta t_i' \stackrel{\text{def}}{=} \Delta t_i - \sum\limits_{j=1}^{c} \ell_{ij} \cdot \Delta s_{ij} > 0$.

To eliminate this discrepancy, we need to repeat the same procedure: divide $\Delta t_i'$ by L_i and again cut down those slownesses that start going outside the corresponding intervals. Because of this cutting down, we may still get some discrepancy remaining, etc.

So, if we apply this idea in a straightforward way, we may need a large number of iterations to fully compensate for the original travel time discrepancy. The need for a large number of iterations leads to a drastic increase in computation time – which, for the seismic inverse problems, is already large.

It is therefore desirable to avoid these iterations and directly come up with a solution which provides the needed compensation of the travel time and at the same time, keeps all the corrected slownesses within the corresponding intervals.

Formulation of the problem in precise terms. For $\Delta t_i > 0$, we would like to find a value $\Delta s > 0$ such that if we take $\Delta s_{ij} = \Delta s$ for all j for which $\Delta s \le \overline{\Delta}_j$ and $\Delta s_{ij} = \overline{\Delta}_j$ for all other j, then we will satisfy the equation (1).

For $\Delta t_i < 0$, we would like to find a value $\Delta s < 0$ such that if we take $\Delta s_{ij} = \Delta s$ for all j for which $\Delta s \ge \underline{\Delta}_j$ and $\Delta s_{ij} = \underline{\Delta}_j$ for all other j, then we will satisfy the equation (1).

Analysis of the problem. In the desired solution, we have $\Delta s_{ij} = \overline{\Delta}_j$ for the values j for which $\overline{\Delta}_j$ is smaller than a certain threshold.

This desired solution is easier to describe if we first soft all the values $\overline{\Delta}_j$ into a non-decreasing sequence

$$\overline{\Delta}_{(1)} \le \overline{\Delta}_{(2)} \le \ldots \le \overline{\Delta}_{(c)}.$$

Then, in the desired solution, there is some index p for which $\Delta s_{i(j)} = \overline{\Delta}_{(j)}$ for all $j \le p$. The common value Δs for the indices $j > p$ can be found from the condition (1), i.e., from the condition that $A_p + \mathcal{L}_p \cdot \Delta s = \Delta t_i$, where we denoted $A_p \stackrel{\text{def}}{=} \sum\limits_{i=1}^{p} \ell_{(i)j} \cdot \overline{\Delta}_{(j)}$ and $\mathcal{L}_p \stackrel{\text{def}}{=} \sum\limits_{j=p+1}^{c} \ell_{i(j)}$. Therefore, we will get $\Delta s = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$.

For the correctly selected index p, all values $\overline{\Delta}_{(j)}$ for which we "cut off" must be smaller than this Δs, and all

the other values $\overline{\Delta}_{(j)}$ must be larger than (or equal to) this Δs. Since the values $\overline{\Delta}_{(j)}$ are sorted in increasing order, it is sufficient to check that $\overline{\Delta}_{(p)} < \Delta s \le \overline{\Delta}_{(p+1)}$.

If for some p, we get $\Delta s > \overline{\Delta}_{(p+1)}$, this means that need to cut some more – otherwise, for $j = p+1$, we will still have the value outside the desired interval. On the other hand, if we get $\Delta s \le \overline{\Delta}_{(p)}$, then there was no reason to cut off at p-th level – so we need to cut less.

Designing an algorithm. This analysis can be naturally be translated into an algorithm. First, we sort the values $\overline{\Delta}_j$; sorting takes time $O(c \cdot \log(c))$; see, e.g., [7]. Then, for every p from 0 to n, we compute the value $\Delta s = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$ and check whether $\overline{\Delta}_{(p)} < \Delta s \le \overline{\Delta}_{(p+1)}$. Once we know A_p, computing A_{p+1} requires just one step – since we need to add one term to the sum. Thus, we to compute all c such values, we need $O(c)$ steps – to the total of $O(c \cdot \log(c)) + O(c) = O(c \cdot \log(c))$. So, we arrive at the following algorithm.

Resulting algorithm. It is sufficient to describe the case when $\Delta t_i > 0$ (the case when $\Delta t_i < 0$ is treated similarly). In this case, we first sort all c values $\overline{\Delta}_j$ along the i-th path into a non-decreasing sequence

$$\overline{\Delta}_{(1)} \le \overline{\Delta}_{(2)} \le \ldots \le \overline{\Delta}_{(c)}.$$

Then, for every p from 0 to c, we compute the values A_p and $\mathcal{L}_p$ as follows: $A_0 = 0$, $\mathcal{L}_0 = L_i$,

$$A_p = A_{p-1} + \ell_{i(p)} \cdot \overline{\Delta}_{(p)}, \;\; \mathcal{L}_p = \mathcal{L}_{p-1} - \ell_{i(p)}.$$

After that, for each p, we compute $\Delta s = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$ and check whether $\overline{\Delta}_{(p)} < \Delta s \le \overline{\Delta}_{(p+1)}$. Once this condition is satisfied, we take $\Delta s_{i(j)} = \overline{\Delta}_{(j)}$ for $j \le p$, and $\Delta s_{i(j)} = \Delta s$ for $j > p$.

When $\Delta t_i < 0$, we similarly sort the values $\underline{\Delta}_j$ into a decreasing sequence, and find p so that the first p corrections are "maxed out" to $\underline{\Delta}_j$, and the rest $c - p$ corrections are determined from the condition $\Delta s = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$.

Comment. Once we have computed these corrections for all the paths, then for each cell j, we take the average (or weighted average) of all the corrections coming from all the paths which pass through this cell.

Example showing efficiency (and feasibility) of the new approach. Let us consider a simple example of two vertical layers of height d (see above picture), with $s > s'$. We assume that the structure below the second layer is so heavy that all the signals simply bounce back from the bottom of the second layer (in real geological situations, this is what happens, e.g., at the Moho surface). For simplicity, we consider only one signal.

Usually, the closer to the surface, the more information we have about the layer. In this example, we assume that we know s exactly, but we only know an approximate value $\widetilde{s}'$ for s' ($\Delta s' \stackrel{\text{def}}{=} \widetilde{s}' - s' \ne 0$). We start with the known values s and $\widetilde{s}'$ and perform iterations following both the original Hole's algorithm and the new interval method.

When the angles φ and φ' are small ($\varphi \ll 1$, $\varphi' \ll 1$), then $\sin(\varphi) \approx \varphi$, $\sin(\varphi') \approx \varphi'$, and we can analytically trace the computations; for details, see [3]. For example, the horizontal distance between the source and the sensor is $2d \cdot (\tan(\varphi) + \tan(\varphi')) \approx 2d \cdot (\varphi + \varphi')$.

In the original Hole's algorithm, the discrepancy in the travel times is uniformly divided between the whole path. As a result, we replace the original approximate slowness $\widetilde{s}' = s' + \Delta s'$ with a more accurate estimate $s' + \dfrac{\Delta s'}{2}$. Hence, the approximation error decreases by a factor of 2. So, e.g., in 7 iterations, we can reduce this error to $< 1\%$ level.

In the new method, we take into account that the value s is already known, i.e., that it is within the given interval $[s, s]$. In this case, the entire discrepancy is corrected by changing only the value s'. Hence, we get the correct value s' in a single iteration.

4. Case of Interval Prior Knowledge: A New Linear Time Algorithm

Motivation: a linear-time algorithm exists for a similar problem of minimizing variance without linear constraints. As we have mentioned, the original Hole's code formulas are related to minimize the variance under a linear constraint (1).

In general, the problem of minimizing variance under interval uncertainty has many other practical applications beyond geophysics. (The only difference is that in most applications, there is no linear constraint similar to (1)). In particular, this general problem has application in geophysics [22, 23].

For this general problem, we have also proposed an $O(c \cdot \log(c))$ algorithm; see, e.g., [15, 16] and references therein.

Recently, we have designed a new algorithm that computes the desired minimum in linear time $O(c)$ [27]. In this paper, we show that a similar linear-time algorithm can be proposed for the case when we want to minimize the variance under an additional linear constraint.

An auxiliary algorithm behind the existing linear-time algorithm. The linear-time algorithm from [27] is based

on the known fact that we can compute the median of a set of n elements in linear time; see, e.g., [7].

The use of median in this algorithm is similar to the one from [6, 10].

A new linear-time algorithm. The proposed algorithm is iterative. At each iteration of this algorithm, we have three sets:

- the set J^- of all the indices j from 1 to c for which we already know that in the desired solution, the corresponding value Δs_{ij} will be cut off (i.e., $\Delta s_{ij} = \overline{\Delta}_j$);
- the set J^+ of all the indices j for which we already know that in the desired solution, the corresponding value Δs_{ij} will *not* be cut off (i.e., $\Delta s_{ij} < \overline{\Delta}_j$);
- the set $J = \{1, \ldots, c\} - J^- - J^+$ of the indices j for which we are still undecided.

In the beginning, $J^- = J^+ = \emptyset$ and $J = \{1, \ldots, c\}$. At each iteration, we also update the values of two auxiliary quantities $A^- \stackrel{\text{def}}{=} \sum\limits_{j \in J^-} \ell_{ij} \cdot \overline{\Delta}_j$ and $\mathcal{L}^+ \stackrel{\text{def}}{=} \sum\limits_{j \in J^+} \ell_{ij}$. In principle, we could compute these values by computing these sums, but to speed up computations, on each iteration, we update these two auxiliary values in a way that is faster than re-computing the corresponding two sums. Initially, since $J^- = J^+ = \emptyset$, we take $A^- = \mathcal{L}^+ = 0$.

At each iteration, we do the following:

- first, we compute the median m of the set J (median in terms of sorting by $\overline{\Delta}_j$);
- then, by analyzing the elements of the undecided set J one by one, we divide them into two subsets

$$P^- \stackrel{\text{def}}{=} \{j : \overline{\Delta}_j \le \overline{\Delta}_m\}, \quad P^+ \stackrel{\text{def}}{=} \{j : \overline{\Delta}_j > \overline{\Delta}_m\};$$

- we compute $a^- \stackrel{\text{def}}{=} A^- + \sum\limits_{j \in P^-} \ell_{ij} \cdot \overline{\Delta}_j$ and

$$\ell^+ \stackrel{\text{def}}{=} \mathcal{L}^+ + \sum_{j \in P^+} \ell_{ij};$$

- then, we compute $\Delta s = \dfrac{\Delta_i - a^-}{\ell^+}$; also, among all the values from P^+, we select the smallest value, which we will denote by $\overline{\Delta}_{(p+1)}$;
- if $\Delta s > \overline{\Delta}_{(p+1)}$, then we replace J^- with $J^- \cup P^-$, A^- with a^-, and J with P^+;
- if $\Delta s \le \overline{\Delta}_m$, then we replace J^+ with $J^+ \cup P^+$, $\mathcal{L}^+$ with ℓ^+, and J with P^-;
- finally, if $\overline{\Delta}_m < \Delta s \le \overline{\Delta}_{(p+1)}$, then we replace J^- with $J^- \cup P^-$, J^+ with $J^+ \cup P^+$, and J with $\emptyset$.

At each iteration, the set of undecided indices is divided in half. Iterations continue until all indices are decided, after which we return $\Delta s_{ij} = \overline{\Delta} s_j$ when $\overline{\Delta}_j \le \overline{\Delta}_m$ and $\Delta s_{ij} = \Delta s$ otherwise.

Proof that the new algorithm for computing $\overline{V}$ requires linear time. At each iteration, computing median requires linear time, and all other operations with J require time t linear in the number of elements $|J|$ of J: $t \le C \cdot |J|$ for some constant C. We start with the set J of size c; on the next iteration, we have a set of size $c/2$, then $c/4$, etc. Thus, the overall computation time is $\le C \cdot (c + c/2 + c/4 + \ldots) \le C \cdot 2c$, i.e., linear in c.

5. Case of Fuzzy Prior Knowledge

Main idea. As we have mentioned, one of the reasons why the mathematically valid solution is not geophysically meaningful is that at some points, the velocity is outside the interval of values which are possible at this depth for this particular geological region.

Additional information provided by experts: general case. To take this expert knowledge into consideration, it is reasonable to explicitly solicit, from the experts, the information about possible values of slownesses – and then modify the inverse algorithms in such a way that the velocities are consistent with this knowledge.

Specifically, for each cell j, a geophysicist provides us with his estimate of possible values of the corresponding slowness s_j. As we have mentioned, an expert often describes this information by using words from the natural language, like "most probably, the value of slowness is within 6 and 7, but it is somewhat possible to have values between 5 and 8". To formalize this knowledge, it is natural to use fuzzy set theory, a formalism specifically designed for describing this type of informal ("fuzzy") knowledge; see, e.g., [4, 8, 14, 21]

As a result, for every cell j, we have a fuzzy set $\mu_j(s)$ which describes the expert's prior knowledge about s_j. For every cell j and for each possible value s_j, the number $\mu_j(s_j)$ describes the expert's degree of certainty that s_j is a possible value of the corresponding slowness.

An alternative user-friendly way to represent a fuzzy set is by using its α-cuts $\{s \,|\, \mu_j(s) > \alpha\}$ (or $\{s \,|\, \mu_j(s) \ge \alpha\}$); see, e.g., [5, 14, 19, 20, 21]. For example, the α-cut corresponding to $\alpha = 0$ is the set of all the values which are possible at all, the α-cut corresponding to $\alpha = 0.1$ is the set of all the values which are possible with degree of certainty at least 0.1, etc. In these terms, a fuzzy set can be viewed as

a nested family of intervals $[\underline{s}_j(\alpha), \overline{s}_j(\alpha)]$ corresponding to different level α.

Comment. For some cells – e.g., in some cells which are close to the surface and for which the geophysical structure is well known – we may even know the *exact* values s_j of slowness. Since a crisp number is a particular case of a fuzzy set, this information can also be expressed in fuzzy terms – by saying that for each of these cells, the geophysicist provides us with a crisp set $\{s_j\}$.

In terms of α-cuts, this means that for every degree α, the corresponding intervals are degenerate intervals $[s_j, s_j]$.

How to use this expert knowledge in solving the seismic inverse problem: precise formulation of the corresponding optimization problem. In general, the solution $(s_1, s_2, \ldots)$ is satisfactory if s_1 is a possible value of slowness in the first cell, and s_2 is a possible value of slowness in the second cell, etc. The corresponding membership functions $\mu_j(s_j)$ describe to what extent s_j is the possible value of slowness in the j-th cell. So, if we use the simplest possible $\min$ operation to describe "and", we conclude that the degree with which a solution is satisfactory can be described by the value $\min(\mu_1(s_1), \mu_2(s_2), \ldots)$.

When we solve the inverse problem, it is reasonable to look for a solution for which this degree of satisfaction is the largest possible: $\min(\mu_1(s_1), \mu_2(s_2), \ldots) \to \max$.

How can we solve this problem: reduction to the case of interval uncertainty. Maximizing the overall degree of satisfaction means that we want to find the largest value α for which $\mu_j(s_j) \geq \alpha$ for all j, i.e., for which, for every cell j, the slowness s_j belongs to the corresponding interval $[\underline{s}_j(\alpha), \overline{s}_j(\alpha)]$.

For each α, we thus face an auxiliary *interval uncertainty problem:* for each cell, we know the corresponding interval, and we want to find a solution to the seismic inverse problem for which all the slownesses are within the corresponding intervals. It is worth mentioning that this interval problem can be of separate practical interest: it is a particular case of the fuzzy uncertainty problem corresponding to the case when the only information coming from an expert is an interval $[\underline{s}_j, \overline{s}_j]$ of possible value of each slowness s_j.

Once we know how to solve this interval problem, we can easily solve the original fuzzy problem as follows. For each $\alpha = 0$, $\alpha = 0.1$, $\alpha = 0.2$, etc., we solve the interval problem with the corresponding intervals $[\underline{s}_j(\alpha), \overline{s}_j(\alpha)]$. Eventually, we will reach such a value of α that the process no longer converges – so the inverse problem with these too narrow interval restriction does not have a solution. Then, the solution corresponding to the previous value α – i.e., to the largest value α for which the process converged – is returned as the desired solution to the seismic inverse problem.

6. Case of Probabilistic Prior Knowledge

Often, prior information comes from processing previous observations of the region of interest. In this case, before our experiments, for each cell j, we know a prior (approximate) slowness value $\widetilde{s}_j$, and we know the accuracy (standard deviation) σ_j of this approximate value $\widetilde{s}_j$. It is known that this prior information can lead to much more accurate velocity models; see, e.g., [18]. How can we modify Hole's algorithm so that it takes this prior information into account?

Due to the prior knowledge, for each cell j, the ratio $\dfrac{(s_j^{(k)} + \Delta s_{ij}) - \widetilde{s}_j}{\sigma_j}$ is normally distributed with 0 mean and variance 1. Since each path i consists of a reasonable number of cells, we can thus conclude that the sample variance of this ratio should be close to σ_j, i.e., that

$$\frac{1}{n} \cdot \sum_{j=1}^{c} \frac{((s_j^{(k)} + \Delta s_{ij}) - \widetilde{s}_j)^2}{\sigma_j^2} = 1. \tag{4}$$

So, to find the corrections Δs_{ij}, we must minimize the objective function (variance)

$$V \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{j=1}^{c} \Delta s_{ij}^2 - \left(\frac{1}{n} \cdot \sum_{j=1}^{c} \Delta s_{ij} \right)^2. \tag{5}$$

under the constraints (1) and (4).

By applying the Lagrange multiplier method to this problem, we can reduce this problem to the unconstrained minimization problem

$$\frac{1}{n} \cdot \sum_{j=1}^{c} \Delta s_{ij}^2 - \left(\frac{1}{n} \cdot \sum_{j=1}^{c} \Delta s_{ij} \right)^2 +$$

$$\lambda \cdot \left(\sum_{j=1}^{c} \ell_{ij} \cdot \Delta s_{ij} - \Delta t_i \right) +$$

$$\mu \cdot \frac{1}{n} \cdot \sum_{j=1}^{c} \frac{(s_j^{(k)} + \Delta s_{ij} - \widetilde{s}_j)^2}{\sigma_j^2} \to \min. \tag{6}$$

Differentiating this equation by Δs_{ij} and equating the derivative to 0, we conclude that

$$\frac{2}{n} \cdot \Delta s_{ij} - \frac{2}{n} \cdot \overline{\Delta s} + \lambda \cdot \ell_{ij} + \frac{2\mu}{n \cdot \sigma_j^2} \cdot (s_j^{(k)} + \Delta s_{ij} - \widetilde{s}_j) = 0,$$

where

$$\overline{\Delta s} \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{j=1}^{c} \Delta s_{ij}. \tag{7}$$

Once we fix λ, μ, and $\overline{\Delta s}$, we get an explicit expression for the values Δs_{ij}. Substituting these expressions into the equations (1), (4), and (7), we get an easy-to-solve system of 3 non-linear equations with 3 unknowns, which we can solve, e.g., by using Newton's method.

Now, instead of explicit formulas for a transition from $s_j^{(k)}$ to $s_j^{(k+1)}$, we need a separate iteration process – so the computation time is somewhat larger, but we get a more geophysically meaningful velocity map – that takes prior knowledge into account.

7. Case of Multiple-Type Prior Knowledge

Practical need for prior knowledge: reminder. In many real-life problems, it is difficult or even impossible to directly measure the desired physical quantities. In such situations, we measure other quantities, which are related to the desired ones by known formulas, and then reconstruct the values of the desired quantities from these measurement results.

The reconstructed values of the desired quantities are sometimes outside the range of what an expert would consider reasonable. In such situations, it is desirable to describe the expert's knowledge (about what is reasonable) as a precisely formulated constraint on the desired values, and to incorporate these constraints into the reconstruction process.

In the previous sections, we have shown that different types of expert knowledge can be naturally formalized in interval, fuzzy, and probabilistic terms. We also showed, on the example of the seismic inverse problem, how each of these types of expert knowledge can be used in the solution process.

Practical need for multiple-type prior knowledge. Previously, we (implicitly) assumed that we have only one type of expert knowledge – e.g., only interval knowledge, or only fuzzy knowledge, etc. In some practical situations, however, we may have multiple-type expert knowledge: e.g., one expert provides interval bounds, another expert provides probabilistic knowledge, etc.

This multiple-type prior knowledge is especially important for *cyberinfrastructure*. The main objective of cyberinfrastructure is to be able to seamlessly move data between different databases (where this data is stored in different formats), to feed the combined data into a remotely located program (which may require yet another data format), and to return the result to the user; see, e.g., [1, 12, 25]. It is also important to gauge the quality and accuracy of this result. We often have different models for describing uncertainty of different databases and programs; it is therefore important to be able to consider multiple-type prior knowledge; see, e.g., [9, 17].

How to use multiple-type prior knowledge in the seismic inverse problem. We have mentioned that in the traditional approach, we minimize (5) under the constraint (1). Different types of prior knowledge mean adding constraints on Δs_{ij}. Probabilistic prior knowledge is naturally formalized as a constraint (4), and interval prior knowledge is naturally formalized as a constraint (2). Thus, when both probabilistic and interval prior knowledge are present, we must minimize (5) under the constraints (1), (2), and (4).

If we replace the equality in (4) by an inequality (≤ 1 instead of $= 1$), then we get a problem of minimizing a convex function under convex constraints, a problem for which there are known efficient algorithms; see, e.g., [26].

For example, we can use a method of alternating projections, in which we first add a correction that satisfy the first constraint, then the additional correction that satisfies the second constraint, etc. In our case, we first add equal values of Δs_{ij} to satisfy the constraint (5), then we restrict the values to the nearest points from the interval $[\underline{s}_j, \overline{s}_j]$ – to satisfy the constraint (2), and after that, find the extra corrections that satisfy the condition (4), after which we repeat the cycle again until the process converges.

8 Conclusion

The paper deals with the difficult *seismic inverse problem*, in which a 3-D field (velocities of the seismic waves) has to be reconstructed. The classical approach is to transform this problem into a huge non-linear system of equation and to use iterative techniques to solve the problem. Often, the classical approach leads to solutions that are not realistic. However, the expert has an idea of what he should not get and he can express this idea as a set of constraints. The main contribution of the paper is to add these additional knowledge, given by the expert, to the classical approach, inside the iterative method.

Acknowledgments. This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Star Award from the University of Texas System, and Texas Department of Transportation grant No. 0-5453.

The authors are thankful to participants of SCAN'06 for valuable discussions and to the anonymous referees for useful suggestions.

References

[1] R. Aldouri, G. R. Keller, A. Gates, J. Rasillo, L. Salayandia, V. Kreinovich, J. Seeley, P. Taylor, and S. Holloway. GEON: Geophysical data add the 3rd dimension in geospatial studies. In: *Proceedings of the ESRI International User Conference 2004*, San Diego, California, August 9–13, 2004, Paper 1898.

[2] M. G. Averill, K. C. Miller, G. R. Keller, V. Kreinovich, R. Araiza, S. A. Starks. Using expert knowledge in solving the seismic inverse problem. In: *Proceedings of the 24nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2005*, Ann Arbor, Michigan, June 22–25, 2005, pp. 310–314.

[3] M. G. Averill, K. C. Miller, G. R. Keller, V. Kreinovich, R. Araiza, and S. A. Starks. Using expert knowledge in solving the seismic inverse problem. *International Journal of Approximate Reasoning* (to appear).

[4] G. Bardossy and J. Fodor. *Evaluation of Uncertainties and Risks in Geology*. Springer Verlag, Berlin, 2004.

[5] G. Bojadziev and M. Bojadziev. *Fuzzy Sets, Fuzzy Logic, Applications*, World Scientific, Singapore, 1995.

[6] P. van der Broek and J. Noppen. Fuzzy weighted average: alternative approach. In: *Proceedings of the 25th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2006*, Montreal, Quebec, Canada, June 3–6, 2006.

[7] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[8] R. Demicco and J. Klir (Eds.). *Fuzzy Logic in Geology*. Academic Press, New York, 2003.

[9] A. Gates, V. Kreinovich, L. Longpré, P. Pinheiro da Silva, and G. R. Keller. Towards secure cyberinfrastructure for sharing border information. In: *Proceedings of the Lineae Terrarum: International Border Conference*, El Paso, Las Cruces, and Cd. Juárez, March 27–30, 2006.

[10] P. Hansen, M. V. P. de Aragao, and C. C. Ribeiro. Hyperbolic 0-1 programming and optimization in information retrieval. *Math. Programming*. 52:255–263, 1991.

[11] J. A. Hole. Nonlinear high-resolution three-dimensional seismic travel time tomography. *J. Geophysical Research* 97:6553–6562, 1992.

[12] G. R. Keller, T. G. Hildenbrand, R. Kucks, M. Webring, A. Briesacher, K. Rujawitz, A. M. Hittleman, D. R. Roman, D. Winester, R. Aldouri, J. Seeley, J. Rasillo, R. Torres, W. J. Hinze, A. Gates, V. Kreinovich, and L. Salayandia. A community effort to construct a gravity database for the United States and an associated Web portal", In: A. K. Sinha (ed.), *Geoinformatics: Data to Knowledge*, Geological Society of America Publ., Boulder, Colorado, 2006, pp. 21–34.

[13] G. R. Keller, S. A. Starks, A. Velasco, M. Averill, R. Araiza, G. Xiang, and V. Kreinovich. Towards combining probabilistic, interval, fuzzy uncertainty, and constraints: on the example of inverse problem in geophysics. In: *Proceedings of the Second International Conference on Fuzzy Sets and Soft Computing in Economics and Finance FSSCEF'2006*, St. Petersburg, Russia, June 28–July 1, 2006, pp. 47–54.

[14] G. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall, New Jersey, 1995.

[15] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos. Interval versions of statistical techniques, with applications to environmental analysis, bioinformatics, and privacy in statistical databases. *Journal of Computational and Applied Mathematics*, 199(2):418–423, 2007.

[16] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpré, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos. Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity. *Reliable Computing*, 12(6):471–501, 2006.

[17] L. Longpré and V. Kreinovich. How to efficiently process uncertainty within a cyberinfrastructure without sacrificing privacy and confidentiality. In: N. Nedjah, A. Abraham, and L. de Macedo Mourelle (Eds.), *Computational Intelligence in Information Assurance and Security*, Springer-Verlag, Berlin-Heidelberg (to appear).

[18] M. Maceira, S. R. Taylor, C. J. Ammon, X. Yang, A. Velasco. High-resolution Rayleigh wave slowness tomography of Central Asia. *Journal of Geophysical Research*, Vol. 110, paper B06304, 2005.

[19] R. E. Moore and W. A. Lodwick. Interval analysis and fuzzy set theory. *Fuzzy Sets and Systems*, 135(1):5–9, 2003.

[20] H. T. Nguyen and V. Kreinovich. Nested intervals and sets: concepts, relations to fuzzy sets, and applications. In: R. B. Kearfott and V. Kreinovich (eds.). *Applications of Interval Computations*, Kluwer, Dordrecht, 1996, pp. 245–290.

[21] H. T. Nguyen and E. A. Walker. *A First Course in Fuzzy Logic*. CRC Press, Boca Raton, Florida, 2005.

[22] P. Nivlet, F. Fournier, and J. Royer. A new methodology to account for uncertainties in 4-D seismic interpretation. In: *Proc. 71st Annual Int'l Meeting of Soc. of Exploratory Geophysics SEG'2001*, San Antonio, TX, September 9–14, 2001, 1644–1647.

[23] P. Nivlet, F. Fournier, and J. Royer. Propagating interval uncertainties in supervised pattern recognition for reservoir characterization. In: *Proc. 2001 Society of Petroleum Engineers Annual Conf. SPE'2001*, New Orleans, LA, September 30–October 3, 2001, paper SPE-71327.

[24] R. L. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton, New Jersey, 1994.

[25] A. K. Sinha (ed.), *Geoinformatics: Data to Knowledge*, Geological Society of America Publ., Boulder, Colorado, 2006.

[26] S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

[27] G. Xiang, M. Ceberio, and V. Kreinovich. *Computing Population Variance and Entropy under Interval Uncertainty: Linear-Time Algorithms*. University of Texas at El Paso, Department of Computter Science, Technical Report UTEP-CS-06-28b, November 2006.

[28] C. A. Zelt and P. J. Barton. Three-dimensional seismic refraction tomography: A comparison of two methods applied to data from the Faeroe Basin. *J. Geophysical Research* 103:7187–7210, 1998.

Adding Constraints to Situations When, In Addition to Intervals, We Also Have Partial Information about Probabilities

Martine Ceberio
Vladik Kreinovich
Gang Xiang
Dept. of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
contact vladik@utep.edu

Scott Ferson
Applied Biomathematics
100 North Country Road
Setauket, NY 11733, USA
scott@ramas.com

Cliff Joslyn
Distributed Knowl. Syst. Team
Computer Research Group
Los Alamos National Lab
Mail Stop B265
Los Alamos, NM 87545, USA
joslyn@lanl.gov

Abstract

In many practical situations, we need to combine probabilistic and interval uncertainty. For example, we need to compute statistics like population mean $E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ *or population variance* $V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2$ *in the situations when we only know intervals* $\mathbf{x}_i$ *of possible values of* x_i*. In this case, it is desirable to compute the range of the corresponding characteristic.*

Some range computation problems are NP-hard; for these problems, in general, only an enclosure is possible. For other problems, there are efficient algorithms. In many practical situations, we have additional information that can be used as constraints on possible cumulative distribution functions (cdfs). For example, we may know that the actual (unknown) cdf is Gaussian. In this paper, we show that such constraints enable us to drastically narrow down the resulting ranges – and sometimes, transform the originally intractable (NP-hard) computational problem of computing the exact range into an efficiently solvable one.

This possibility is illustrated on the simplest example of an NP-problem from interval statistics: the problem of computing the range $\mathbf{V}$ *of the variance* V*.*

We also describe how we can estimate the amount of information under such combined intervals-and-constraints uncertainty.

1. Formulation of the Problem

Statistical analysis is important. Statistical analysis of measurement and observation results is an important part of data processing and data analysis. When faced with new data, engineers and scientists usually start with estimating standard statistical characteristics such as the mean E, the variance V, the cumulative distribution function (cdf) $F(x)$ of each variable, and the covariance and correlation between different variables. In the traditional statistical analysis, we estimate the value of each characteristic by computing the corresponding statistic $C(x_1, \ldots, x_n)$, such as:

- population mean $E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$;
- population variance $V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2$;
- histogram cdf $F_n(x) = \frac{\#i : x_i \le x}{n}$ (where $\#i : P(i)$ denotes "the number of i for which $P(i)$ is true");
- population covariance

$$C_{x,y} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E_x) \cdot (y_i - E_y).$$

Limitations of traditional statistical techniques and the need to consider interval uncertainty. Traditional methods of statistical analysis assume that the measured values $\widetilde{x}_1, \ldots \widetilde{x}_n$ are the actual values $x_1, \ldots, x_n$ of the measured quantities. These methods work well if the variability of each variable is much higher than the measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. For example, the accuracy with which we measure a person's height ($\approx$ 1 cm) is much smaller than the variability in height between different people.

In many practical situations, however, the measurement errors are of the same order of magnitude as variability and

therefore, cannot be ignored. Often, the only information that we have about the measurement errors is that they are upper bounded by Δ_i – and we have no information about the probabilities of different values $\Delta x_i \in [-\Delta_i, \Delta_i]$. In such situations, the only information we have after the measurements about the (unknown) actual value x_i is that x_i belongs to the intervals $\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

In the case of interval uncertainty, instead of the actual (exact) values x_i, we only know the intervals $\mathbf{x}_i$ of possible values of x_i. In this case, we must find the range

$$\mathbf{C} = C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=}$$

$$\{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of possible values of the given statistic.

Adding interval uncertainty to statistical techniques: what is known. There is a lot of research for computing such ranges. Some range computation problems are NP-hard – even the problem of computing the range for the population variance is, in general, NP-hard; see, e.g., [6, 9]. For such problems, in general, we can only compute an enclosure for the desired range.

For other problems, there are efficient algorithms; see, e.g., [2, 6, 9, 11] and references therein. For example, efficient algorithms are possible:

- for computing the range $\mathbf{E}$ of the population mean E,
- for computing the lower endpoints $\underline{V}$ of the range of variance,
- for computing the upper endpoint $\overline{V}$ of the range of variance when the intervals $\mathbf{x}_i$ are not contained in each other, i.e., $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$ for all i and j,
- etc.

Limitations of the existing approach. To explain the main limitation of the existing approach, let us briefly summarize what this approach is doing:

- we start with a statistic $C(x_1, \ldots, x_n)$ for estimating a given characteristic S;
- we evaluate this statistic under interval uncertainty, resulting in $\mathbf{C} = C(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

The main limitation of this idea is that a statistic is only an approximation to the desired statistical characteristic, i.e., $C(x_1, \ldots, x_n) \approx S$. For example, the population mean is only approximately equal to the mean value of the random quantity; similarly, the population variance is only an approximation to the actual variance, etc.

The approximation error $C(x_1, \ldots, x_n) - S \neq 0$ is not always taken into account when we take the interval range $\mathbf{C}$ as the range of the actual values of S.

For example, in this approach, if the values x_i are known exactly, then as a range of the population average, we will get a single number $E = \dfrac{1}{n} \cdot \sum_{i=1}^{n} x_i$ – while in reality, the actual mean can differ from this population average.

Seemingly natural solution can lead to excess width. A natural solution is that, instead of the original statistic C, we consider the bounds C^- and C^+ of the corresponding confidence interval $[C^-, C^+]$.

By definition of the confidence interval, this interval contains the actual value of the characteristic S with an appropriate certainty. For example, under reasonable assumptions (e.g., if the distribution is Gaussian), the interval $[E - k_0 \cdot \sigma, E + k_0 \cdot \sigma]$, where $\sigma \stackrel{\text{def}}{=} \sqrt{V}$ and k_0 (usually, 2, 3, or 6) is a given constant.

Thus, if we compute the interval range $[\underline{C}^-, \overline{C}^-]$ and $[\underline{C}^+, \overline{C}^+]$ for the statistics C^- and C^+, then the corresponding interval $[\underline{C}^-, \overline{C}^+]$ is an enclosure for S (with appropriate certainty). The ranges for C^- and C^+ can indeed be often efficiently computed [1, 5, 6, 9].

The problem with this idea is that a confidence interval is often defined so as to contain the actual value – but not necessarily as the narrowest interval that contains this value. As a result, the interval $[\underline{C}^-, \overline{C}^+]$ may contain excess width.

New idea. Let us instead find the actual range

$$\mathbf{S} = \{S(F) : F \text{ is possible}\}$$

of the characteristic S. Estimating this range is the main problem that we will be solving in this paper.

To solve this main problem, we must be able to solve the following closely related problem: how to describe class $\mathcal{F}$ of all the probability distributions F which are consistent with the given observations $[\underline{x}_i, \overline{x}_i]$?

2. How to Describe Possible Probability Distributions: p-Boxes

Case of exactly known probability distribution. The class of all probability distributions is infinite-dimensional; thus, to exactly describe a probability distribution, we need infinitely many parameters. In a computer, we can only store and process finitely many numbers; thus, if we want to represent probability distributions in a computer, we must select finitely many characteristics that will actually be representing this distribution.

To make this representation useful in practical applications, we must select characteristics which are practically useful. In many practical example, there is a critical threshold x_0 after which some undesirable event happens: a chip delays too much, a panel cracks, etc. In such situations, we want to make sure that the probability of exceeding x_0 is small. The resulting characteristic $\mathrm{Prob}(x_i \le x_0)$ is the value of the cumulative distribution function (cdf) $F(x)$ for $x = x_0$.

Thus, from the practical viewpoint, it is beneficial to describe a probability distribution by its cdf $F(x)$.

Case of partially known probability distribution. When, for every x, we know the exact value of $F(x)$, we thus know the actual probability distribution exactly. So, when we only have partial information about the probability distribution, this means that we do not know the exact values of $F(x)$. Instead, we may know, for every x, an interval $\mathbf{F}(x) = [\underline{F}(x), \overline{F}(x)]$ that contains the actual (unknown) value $F(x)$.

Thus, a natural way to describe partial information about a probability distribution is to describe, for every x, a function $x \to \mathbf{F}(x)$. This function is called a *p-box* [2].

3. Estimates for Statistical Characteristics Based on the Use of p-Boxes

New idea (reminder). We have several observations $x_1, \ldots, x_n$ of a given random variable. These observations may be exact – in which case, we know the exact values of x_i – or, more generally, they may consist of known intervals $\mathbf{x}_i$ which contain the actual (unknown) values x_i of the observed quantity.

Our objective is to estimate the value of a statistical characteristic S based on these observations. Our new idea is that we estimate S in two steps:

- first, we describe the class of all probability distributions which are consistent with the given observations; since we agreed to represent such classes as p-boxes, we must transform observations $x_1, \ldots, x_n$ into a p-box;
- second, we estimate the range of the desired characteristic S based on this p-box.

Kolmogorov-Smirnov (KS) p-box. In statistics, there is a known way to produce bounds on cdfs (i.e., a p-box) from observations: use Kolmorogov-Smirnov (KS) inequalities; see, e.g., [7, 10].

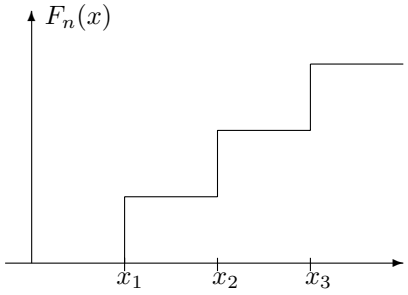
The main idea behind KS inequalities is rather straightforward. Namely, for each x_0, we have

- the actual (unknown) probability $p = F(x_0)$ that $x \le x_0$, and
- the observed frequency $F_n(x_0) = \dfrac{\#i : x_i \le x_0}{n}$.

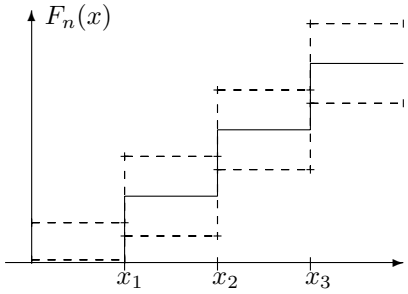
It is known that when n tends to infinity, then the distribution for the frequency tends to normal. Thus, for large n, this distribution is approximately normal. Hence, with given certainty α, we have $p - k \cdot \sigma \le F_n(x_0) \le p + k \cdot \sigma$, where $\sigma = \sqrt{\dfrac{p \cdot (1-p)}{n}}$ is the standard deviation of this simple random variable and $k = k(\alpha)$ is a factor that determines the confidence level. So, with certainty α, we get bounds on $p = F(x_0)$ in terms of $F_n(x_0)$.

We can now use these bounds for $x_0 = x_1$, …, $x_0 = x_n$, and use monotonicity of the cdf $F(x)$ to get bounds $[F_n(x) - \varepsilon, F_n(x) + \varepsilon]$ for all $x \in [x_i, x_{i+1}]$.

Graphically, for a histogram



the Kolmogorov-Smirnov p-box takes the form:



For interval-valued data $[\underline{x}_i, \overline{x}_i]$, instead of single histogram, we have a p-box $[\underline{F}_n(x), \overline{F}_n(x)]$ formed by:

- the histogram $\underline{F}_n(x)$ generated by the values $\overline{x}_1, \ldots, \overline{x}_n$, and
- the histogram $\overline{F}_n(x)$ generated by the values $\underline{x}_1, \ldots, \underline{x}_n$.

To get a guaranteed bound (with appropriate certainty), we perform the same ε-enlargement to this p-box, producing a new p-box

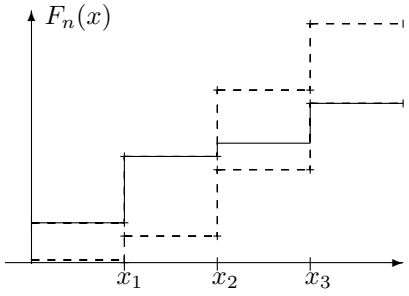
$$\mathbf{F}(x) = [\max(\underline{F}_n(x) - \varepsilon), 0), \min(\overline{F}_n(x) + \varepsilon, 1)].$$

Computing bounds for variance based on the KS p-box. Most known algorithms for computing the lower and upper bounds for the population variance under the interval uncertainty (see, e.g., [6, 9]) use the results of the calculus-type analysis of optimal values. Specifically, we use the following facts:

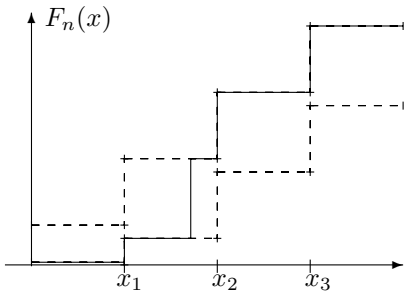
- if the function V attains a maximum or minimum for some value x_i which is strictly inside the interval $[\underline{x}_i, \overline{x}_i]$, then $\dfrac{\partial f}{\partial x_i} = 0$;
- if V attains a maximum for $x_i = \underline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \le 0$;
- if V attains a minimum for $x_i = \underline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \ge 0$;
- if V attains a maximum for $x_i = \overline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \ge 0$;
- if V attains a minimum for $x_i = \overline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \le 0$.

For the actual variance $V = \int x^2\, dF(x) - \left(\int x\, dF(x)\right)^2$, a similar reasonably simple calculus-type analysis leads to the following conclusions:

- the minimum $\underline{V}$ of the variance V is attained when the cdf $F(x) \in \mathbf{F}(x)$ first follows the upper cdf $\overline{F}(x)$, then stays horizonal, and then follows the lower cdf $\underline{F}(x)$:



- the maximum $\overline{V}$ of the variance V is attained when the cdf $F(x) \in \mathbf{F}(x)$ first follows the lower cdf $\underline{F}(x)$, and then jumps (vertically) to the upper cdf $\overline{F}(x)$:



Comment. The only difference with the case of population variance – in which we have finitely many variables $x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n$ – is that now we have an unknown function $F(x) \in \mathbf{F}(x)$ – i.e., in effect, infinitely many variables $F(x) \in \mathbf{F}(x)$ corresponding to different values x.

Computational complexity of computing $\underline{V}$ and $\overline{V}$. For the bounds on the *population* variance:

- we can compute $\underline{V}$ in linear time $O(n)$ [12];
- computing $\overline{V}$ is, in general, NP-hard;
- when $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$, we can compute $\overline{V}$ in linear time [12].

For the actual variance, if we use KS p-box, then the only remaining question is when to make a jump. For n data points, there are n possible interval containing this jump. For each interval, finding the best location is an easy-to-solve (quadratic) optimization problem with one variable, so its complexity does not depend on n. Thus, by applying the above observation:

- we can compute $\underline{V}$ in linear time $O(n)$, and
- we can compute $\overline{V}$ in linear time $O(n)$;

Conclusion. When we go from computing the range of the *population* variance to computing the range of the *actual* variance, we not only make our estimates more adequate – we also, in general, make computations much faster.

4 How to Handle Additional Constraints

Possibility of additional information. In the previous text, we assumed that the only information we have about the cdf $F(x)$ is that it is contained in the given p-box: $F(x) \in \mathbf{F}(x)$. However, often, we have additional information about $F(x)$.

This information that can be used as constraints on possible cdfs $F(x) \in \mathbf{F}(x)$. It is desirable to use these constraints when estimating statistical characteristics – similarly to the way constraints can be combined with traditional interval computations; see, e.g., [3].

Types of additional information. Often, we sometimes know the *shape* of $F(x)$, i.e., we know that $F(x) = F_0(x, a_1, \ldots, a_n)$ for a known function F_0 and for some parameters a_i. Usually, we do not know the *exact* values of each of these parameters; we may know the intervals $\mathbf{a}_i = [\underline{a}_i, \overline{a}_i]$ that contain the actual (unknown) values of these parameters.

A typical situation is when this dependence is linear in a_i, i.e., when

$$F(x) = F_0\left(\sum_{i=1}^{n} a_i \cdot e_i(x)\right).$$

To be more precise, the known dependence may not be linear in terms of the *given* parameters, but it may be described in this form if we use *appropriate* parameters.

For example, if we know that the actual distribution is Gaussian, this means that $F(x) = F_0\left(\dfrac{x-a}{\sigma}\right)$ for some parameters a and σ. With respect to the given parameters a and σ, this dependence is not linear, but if we select new parameters $a_1 \stackrel{\text{def}}{=} \dfrac{1}{\sigma}$ and $a_2 \stackrel{\text{def}}{=} -\dfrac{a}{\sigma}$, then we get the desired linear form: $F(x) = F_0(a_1 \cdot x + a_2)$.

How to take this additional information into account: first seemingly natural solution. We have mentioned that a natural way to represent a class of probability distributions is to find an appropriate p-box. Thus, it seems natural to find a p-box containing this class, i.e., for every x, to find the interval of possible values of $F(x)$ corresponding to the given class.

Once the p-box is found, we can then estimate the range of the desired characteristic – e.g., of the variance V – based on this p-box.

Limitations of the above seemingly natural approach. This approach indeed provided a guaranteed bound (enclosure) for the desired range – but it may also have excess width.

For example, if we start with the family of all normal distributions with 0 average and standard deviation σ from a given interval $[\underline{\sigma}, \overline{\sigma}]$, then the actual mean is always 0. However, as one can easily check, the corresponding p-box has non-zero width; as a result, it contains distribution with non-zero mean – and thus, the enclosure for the mean computed by using this p-box will contain non-zero values.

Towards exact estimates. Once we have a KS p-box $[\underline{F}(x), \overline{F}(x)]$ based on observations, we know that the actual (unknown) cdf $F(x)$ must be within this interval for all x:

$$\underline{F}(x) \le F(x) \le \overline{F}(x).$$

By definition, the KS p-box is obtained from the values at $x = x_i$ via monotonicity: when $x_i < x < x_{i+1}$, we take $\underline{F}(x) = \underline{F}(x_i)$ and $\overline{F}(x) = \overline{F}(x_{i+1})$. Thus, to guarantee that $F(x) \in \mathbf{F}(x)$ for all x, it is sufficient to check that this enclosure occurs for $x = x_1, \ldots, x_n$, i.e., that

$$\underline{F}(x_i) \le F(x_i) \le \overline{F}(x_i).$$

We know that $F(x) = F_0\left(\sum\limits_{i=1}^{n} a_i \cdot e_i(x)\right)$, so we can conclude that

$$\underline{F}(x_i) \le F_0\left(\sum_{i=1}^{n} a_i \cdot e_i(x)\right) \le \overline{F}(x_i).$$

Since the cdf $F_0(x)$ is monotonic, we can apply the inverse function F_0^{-1} to all the sides and get an equivalent inequality:

$$F_0^{-1}(\underline{F}(x_i)) \le \sum_{i=1}^{n} a_i \cdot e_i(x) \le F_0^{-1}(\overline{F}(x_i)). \qquad (1)$$

Thus, if we know the dependence $S(a_1, \ldots, a_n)$ of the desired characteristic S on the parameters a_i, then we can find the range of this characteristic by finding the minimum and the maximum of the corresponding function $S(a_1, \ldots, a_n)$ under the constraints (1) and $\underline{a}_i \le a_i \le \overline{a}_i$.

In particular, if the dependence $S(a_1, \ldots, a_n)$ is linear in a_i, then the problems of finding the minimum $\underline{S}$ and the maximum $\overline{S}$ are linear programming problems – i.e., problems which can be efficiently solved by known feasible algorithms.

Example. In practice, there are examples when the actual dependence $S(a_1, \ldots, a_n)$ is not linear, but this dependence can be reduced to linear by an appropriate transformation.

For example, for the case of the Gaussian distribution, we may be interested in the variance $V = \sigma^2$. In this case, as we have mentioned, $a_1 = \dfrac{1}{\sigma}$, hence $\sigma = \dfrac{1}{a_1}$, and $V = \dfrac{1}{a_1^2}$. This dependence is non-linear; however, this dependence is strictly increasing. Thus:

- finding the minimum of V is equivalent to finding the maximum of a_1 and
- finding the maximum of V is equivalent to finding the minimum of a_1.

The problem of finding the minimum and maximum of a_1 under linear constraints is already a linear programming problem.

Conclusion. The use of additional information about the probability distribution not only eliminates the excess width; it may also transform the originally NP-hard problem of estimating the range of the variance into a feasible one.

5. Gauging Amount of Uncertainty

Formulation of the problem and a seemingly natural solution. Every time we have uncertainty, an important question is how to gauge the amount of uncertainty; see, e.g., [4]. In the traditional statistical approach, the uncertainty in a probability distribution is usually described by Shannon's entropy

$$S = -\int \rho(x) \cdot \log(\rho(x))\, dx,$$

where $\rho(x) = F'(x)$ is the probability density function of this distribution.

We have already mentioned that in the situations when we have partial information about the probability distribution $F(x)$ – e.g., when we only know that $F(x)$ belongs to a non-degenerate p-box $\mathbf{F}(x) = [\underline{F}(x), \overline{F}(x)]$, a reasonable estimate for an arbitrary statistical characteristic S is the range of possible values of S over all possible distributions $F(x) \in \mathbf{F}(x)$.

It therefore seems natural to apply this approach to entropy as well – and return the range of entropy as a gauge of uncertainty of a p-box; see, e.g., [4, 13].

Limitations of the above (seemingly natural) solution. The problem with the above approach is that every non-degenerate p-box includes discrete distributions, i.e., distributions which take discrete values $x_1, \ldots, x_n$ with finite probabilities. For such distributions, Shannon's entropy is $-\infty$.

Thus, for every non-degenerate p-box, the resulting interval $[\underline{S}, \overline{S}]$ has the form $[-\infty, \overline{S}]$. Thus, once the distribution with the largest entropy $\overline{S}$ is fixed, we cannot distinguish between a very narrow p-box or a very thick p-box – in both case, we end up with the same interval $[-\infty, \overline{S}]$.

It is therefore desirable to develop a new approach that would enable us to distinguish between these two cases.

Our idea: go back to the foundations. To design this new characteristic, let us go back to the foundations, check how Shannon came up with his measure of uncertainty, and see how Shannon's derivations can be modified to the case of p-boxes.

Traditional approach to gauging amount of information: reminder. The traditional Shannon's notion of the amount of information is based on defining information as the (average) number of "yes"-"no" (binary) questions that we need to ask so that, starting with the initial uncertainty, we will be able to completely determine the object.

Discrete case, when we have no information about probabilities. Let us start with the simplest situation when we know that we have n possible alternatives $A_1, \ldots, A_n$, and we have no information about the probability (frequency) of different alternatives. Let us show that in this case, the smallest number of binary questions that we need to determine the alternative is indeed $q \stackrel{\text{def}}{=} \lceil \log_2(n) \rceil$.

Comment. The value $\lceil x \rceil$ is the smallest integer which is larger than or equal to x. It is called the *ceiling* of the number x.

After each binary question, we can have 2 possible answers. So, if we ask q binary questions, then, in principle, we can have 2^q possible results. Thus, if we know that our object is one of n objects, and we want to uniquely pinpoint the object after all these questions, then we must have $2^q \geq n$, i.e., $q \geq \log_2(n)$. To complete the derivation, it is let us show that it is sufficient to ask q questions.

Indeed, let's enumerate all n possible alternatives (in arbitrary order) by numbers from 0 to $n-1$, and write these numbers in the binary form. Using q binary digits, one can describe numbers from 0 to $2^q - 1$. Since $2^q \geq n$, we can this describe each of the n numbers by using only q binary digits. So, to uniquely determine the alternative A_i out of n given ones, we can ask the following q questions: "is the first binary digit 0?", "is the second binary digit 0?", etc, up to "is the q-th digit 0?".

Case of a discrete probability distribution. Let us now assume that we also know the probabilities $p_1, \ldots, p_n$ of different alternatives $A_1, \ldots, A_n$. If we are interested in an individual selection, then the above arguments show that we cannot determine the actual alternative by using fewer than $\log(n)$ questions. However, if we have many (N) similar situations in which we need to find an alternative, then we can determine all N alternatives by asking $\ll N \cdot \log_2(n)$ binary questions.

To show this, let us fix i from 1 to n, and estimate the number of events N_i in which the output is i.

This number N_i is obtained by counting all the events in which the output was i, so $N_i = n_1 + n_2 + \ldots + n_N$, where n_k equals to 1 if in k-th event the output is i and 0 otherwise. The average $E(n_k)$ of n_k equals to $p_i \cdot 1 + (1 - p_i) \cdot 0 = p_i$. The mean square deviation $\sigma[n_k]$ is determined by the formula $\sigma^2[n_k] = p_i \cdot (1 - E(n_k))^2 + (1 - p_i) \cdot (0 - E(n_k))^2$. If we substitute here $E(n_k) = p_i$, we get $\sigma^2[n_k] = p_i \cdot (1 - p_i)$. The outcomes of all these events are considered independent, therefore n_k are independent random variables. Hence the average value of N_i equals to the sum of the averages of n_k: $E[N_i] = E[n_1] + E[n_2] + \ldots + E[n_N] = Np_i$. The mean square deviation $\sigma[N_i]$

satisfies a likewise equation $\sigma^2[N_i] = \sigma^2[n_1] + \sigma^2[n_2] + \ldots = N \cdot p_i \cdot (1-p_i)$, so $\sigma[N_i] = \sqrt{p_i \cdot (1-p_i) \cdot N}$.

For big N the sum of equally distributed independent random variables tends to a Gaussian distribution (the well-known *central limit theorem*), therefore for big N, we can assume that N_i is a random variable with a Gaussian distribution. Theoretically a random Gaussian variable with the average a and a standard deviation σ can take any value. However, in practice, if, e.g., one buys a voltmeter with guaranteed 0.1V standard deviation, and it gives an error 1V, it means that something is wrong with this instrument. Therefore it is assumed that only some values are practically possible. Usually a "k-sigma" rule is accepted that the real value can only take values from $a - k \cdot \sigma$ to $a + k \cdot \sigma$, where k is 2, 3, or 4. So in our case we can conclude that N_i lies between $N \cdot p_i - k \cdot \sqrt{p_i \cdot (1-p_i) \cdot N}$ and $N \cdot p_i + k \cdot \sqrt{p_i \cdot (1-p_i) \cdot N}$. Now we are ready for the formulation of Shannon's result.

Comment. In this quality control example the choice of k matters, but, as we'll see, in our case the results do not depend on k at all.

Let a real number $k > 0$ and a positive integer n be given. The number n is called *the number of outcomes*. By a *probability distribution*, we mean a sequence $\{p_i\}$ of n real numbers, $p_i \geq 0$, $\sum p_i = 1$. The value p_i is called a *probability* of i-th event. Let an integer N is given; it is called *the number of events*. By a *result of N events* we mean a sequence r_k, $1 \leq k \leq N$ of integers from 1 to n. The value r_k is called the *result of k-th event*. The total number of events that resulted in the i-th outcome will be denoted by N_i. We say that the result of N events is *consistent* with the probability distribution $\{p_i\}$ if for every i, we have $N \cdot p_i - k \cdot \sigma_i \leq N_i \leq N + k \cdot \sigma_i$, where $\sigma_i \stackrel{\text{def}}{=} \sqrt{p_i \cdot (1-p_i) \cdot N}$. Let's denote the number of all consistent results by $N_{cons}(N)$. The number $\lceil \log_2(N_{cons}(N)) \rceil$ will be called *the number of questions, necessary to determine the results of N events* and denoted by $Q(N)$. The fraction $Q(N)/N$ will be called the *average number of questions.*

Theorem (Shannon; see, e.g., [8]). *When the number of events N tends to infinity, the average number of questions tends to*

$$S(p) \stackrel{\text{def}}{=} -\sum p_i \cdot \log_2(p_i).$$

Case of a continuous probability distribution. After a finite number of "yes"-"no" questions, we can only distinguish between finitely many alternatives. If the actual situation is described by a real number, then, since there are infinitely many different possible real numbers, after finitely many questions, we can only get an approximate value of this number.

Once we fix the accuracy $\varepsilon > 0$, we can talk about the number of questions that are necessary to determine a number x with this accuracy ε, i.e., to determine an approximate value r for which $|x - r| \leq \varepsilon$.

Once an *approximate* value r is determined, possible *actual* values of x form an interval $[r - \varepsilon, r + \varepsilon]$ of width 2ε. Vice versa, if we have located x on an interval $[\underline{x}, \overline{x}]$ of width 2ε, this means that we have found x with the desired accuracy ε: indeed, as an ε-approximation to x, we can then take the midpoint $(\underline{x} + \overline{x})/2$ of the interval $[\underline{x}, \overline{x}]$.

Thus, the problem of determining x with the accuracy ε can be reformulated as follows: we divide the real line into intervals $[x_i, x_{i+1}]$ of width 2ε ($x_{i+1} = x_i + 2\varepsilon$), and by asking binary questions, find the interval that contains x. As we have shown, for this problem, the average number of binary question needed to locate x with accuracy ε is equal to $S = -\sum p_i \cdot \log_2(p_i)$, where p_i is the probability that x belongs to i-th interval $[x_i, x_{i+1}]$.

In general, this probability p_i is equal to $\int_{x_i}^{x_{i+1}} \rho(x)\, dx$, where $\rho(x)$ is the probability distribution of the unknown values x. For small ε, we have $p_i \approx 2\varepsilon \cdot \rho(x_i)$, hence $\log_2(p_i) = \log_2(\rho(x_i)) + \log_2(2\varepsilon)$. Therefore, for small ε, we have

$$S = -\sum \rho(x_i) \cdot \log_2(\rho(x_i)) \cdot 2\varepsilon - \sum \rho(x_i) \cdot 2\varepsilon \cdot \log_2(2\varepsilon).$$

The first sum in this expression is the integral sum for the integral $S(\rho) \stackrel{\text{def}}{=} -\int \rho(x) \cdot \log_2(x)\, dx$ (this integral is called the *entropy* of the probability distribution $\rho(x)$); so, for small ε, this sum is approximately equal to this integral (and tends to this integral when $\varepsilon \to 0$). The second sum is a constant $\log_2(2\varepsilon)$ multiplied by an integral sum for the interval $\int \rho(x)\, dx = 1$. Thus, for small ε, we have

$$S \approx -\int \rho(x) \cdot \log_2(x)\, dx - \log_2(2\varepsilon).$$

So, the average number of binary questions that are needed to determine x with a given accuracy ε, can be determined if we know the entropy of the probability distribution $\rho(x)$.

Case of p-boxes: description of the situation. Our main motivation is that the traditional approach of interval-valued entropy does not allow us to distinguish between narrow and wide p-boxes. For a wide p-box, it is OK to make a wide interval like $[-\infty, \overline{S}]$, but for narrow p-boxes, we would like to have narrower estimates. Let us therefore consider narrow p-boxes.

Since entropy is defined for smooth (differentiable) cdfs $F(x)$, it is reasonable to start with the case when the central function of a p-box is also smooth. In other words, we

consider p-boxes of the type

$$\mathbf{F}(x) = [F_0(x) - \Delta F(x), F_0(x) + \Delta F(x)],$$

where $F_0(x)$ is differentiable, with derivative $\rho_0(x) \stackrel{\text{def}}{=} F_0'(x)$, and $\Delta F(x)$ is small.

Formulation of the problem. For each $\varepsilon > 0$ and for each distribution $F(x) \in \mathbf{F}(x)$, we can use the above formulas to estimate the average number $S_\varepsilon(F)$ of "yes"-"no" question that we need to ask to determine the actual value with accuracy ε. Our objective is to compute the range

$$[\underline{S}, \overline{S}] = \{S_\varepsilon(F) : F \in \mathbf{F}\}.$$

Known result. It is known (see, e.g., [8]) that asymptotically,

$$\overline{S} \sim -\int \rho_0(x) \cdot \log_2(\rho_0(x))\, dx - \log_2(2\varepsilon).$$

New result. Our new result is that

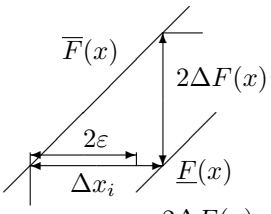
$$\underline{S} \sim -\int \rho_0(x) \cdot \max(2\Delta F(x), 2\varepsilon \cdot \rho_0(x))\, dx.$$

Comment. This result holds when ε and the width of ΔF both tends to 0. If instead we fix the width ΔF and let $\varepsilon \to 0$, then $\overline{S} \to \infty$ but $\underline{S}$ remains finite.

Idea of the proof. When we discretize the distribution, we get $p_i \approx \rho_0(x_i) \cdot \Delta x_i$, hence

$$-\sum p_i \cdot \log_2(p_i) \approx -\int \rho_0(x) \cdot \log(\rho_0(x) \cdot \Delta x)\, dx.$$

To minimize the entropy, we can take the discrete distribution with values $x_1, \ldots, x_n$ as far away from each other as possible. A distribution which is located at x_i and x_{i+1} and has 0 probability to be in between is described by a cdf $F(x)$ which is horizontal on $[x_i, x_{i+1}]$. Thus, we must select a cdf $F(x) \in \mathbf{F}(x)$ for which these horizontal segments are as long as possible. The length of a horizontal segment is bounded by the geometry of the p-box:



Thus, this length cannot exceed $\dfrac{2\Delta F(x)}{\rho_0(x)}$. If this length is $> 2\varepsilon$, then we can take this interval between the sequential values x_i. If this length is $< 2\varepsilon$, then we can still take $\Delta x_i = 2\varepsilon$. Thus, in general, we take $\Delta x_i = \max\left(\dfrac{2\Delta F(x)}{\rho_0(x)}, 2\varepsilon\right)$. Substituting this expression into the above asymptotic formula, we get the desired asymptotic for $\underline{S}$.

6 Conclusions

This paper mainly deals with the following problem: When we have several observations of given random variables, these observations being known as intervals, how is it possible to compute enclosures for some characteristics such as mean values or standard deviation of the true population.

In our paper, we describe feasible algorithms for solving several practically important cases of this problem.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Star Award from the University of Texas System, and Texas Department of Transportation grant No. 0-5453.

The authors are thankful to participants of SCAN'06 for valuable discussions, and to the anonymous referees for important suggestions.

References

[1] E. Dantsin, A. Wolpert, M. Ceberio, G. Xiang, and V. Kreinovich. Detecting outliers under interval uncertainty: a new algorithm based on constraint satisfaction. In: *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'06*, Paris, France, July 2–7, 2006, pp. 802–809.

[2] S. Ferson. *RAMAS Risk Calc 4.0*. CRC Press, Boca Raton, Florida, 2002.

[3] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer-Verlag, London, 2001.

[4] G. J. Klir. *Uncertainty and Information: Foundations of Generalized Information Theory*. J. Wiley, Hoboken, New Jersey, 2005.

[5] V. Kreinovich, L. Longpré, P. Patangay, S. Ferson, and L. Ginzburg. Outlier detection under interval uncertainty: algorithmic solvability and computational complexity. *Reliable Computing*, 11(1):59–76, 2005.

[6] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos. Interval versions of statistical techniques, with applications to environmental analysis, bioinformatics, and privacy in statistical databases. *Journal of Computational and Applied Mathematics*, 199(2):418–423, 2007.

[7] V. Kreinovich, E. J. Pauwels, S. Ferson, and L. Ginzburg. A feasible algorithm for locating concave and convex zones of interval data and its use in statistics-based clustering. *Numerical Algorithms* 37:225–232, 2004.

[8] V. Kreinovich, G. Xiang, and S. Ferson. How the concept of information as average number of "yes-no" questions (bits) can be extended to intervals, p-boxes, and more general uncertainty. In: *Proceedings of the 24th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2005*, Ann Arbor, Michigan, June 22–25, 2005, pp. 80–85.

[9] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpré, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos. Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity. *Reliable Computing*, 12(6):471–501, 2006.

[10] H. M. Wadsworth, Jr. (ed.). *Handbook of statistical methods for engineers and scientists*. McGraw-Hill Publishing Co., New York, 1990.

[11] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman & Hall, New York, 1991.

[12] G. Xiang, M. Ceberio, and V. Kreinovich. *Computing Population Variance and Entropy under Interval Uncertainty: Linear-Time Algorithms*. University of Texas at El Paso, Department of Computter Science, Technical Report UTEP-CS-06-28b, November 2006.

[13] G. Xiang, O. Kosheleva, and G. J. Klir. Estimating information amount under interval uncertainty: algorithmic solvability and computational complexity. In: *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'06*, Paris, France, July 2–7, 2006, pp. 840–847.

Fast and Accurate Multi-Argument Interval Evaluation of Polynomials

Andreas Frommer and Bruno Lang
Fachbereich Mathematik und Naturwissenschaften
Bergische Universität Wuppertal
D-42097 Wuppertal, Germany
{frommer,lang}@math.uni-wuppertal.de

Abstract

The verification of the existence of certain spherical t-designs involves the evaluation of a degree-t polynomial J_t at a very large number of (interval) arguments. To make the overall verification process feasible computationally, this evaluation must be fast, and the enclosures for the function values must be affected with only modest over-estimation. We discuss several approaches for multi-argument interval evaluation of the polynomial J_t and show how they can be adapted to other polynomials p. One particularly effective new method is based on expanding the polynomial p around several points ξ_j and truncating each resulting expansion p_{ξ_j} to a lower-degree polynomial.

1. Introduction

The task of evaluating a polynomial at a large number of arguments occurs repeatedly during the verification of the existence of certain spherical t-designs [2]. An N-point 3D spherical t-design consists of N points $\boldsymbol{x}_i$ on the unit sphere $S^2 \subset \mathbb{R}^3$ such that the quadrature formula

$$\int_{S^2} p(\boldsymbol{x}) d\mu \approx \frac{4\pi}{N} \cdot \sum_{i=1}^{N} p(\boldsymbol{x}_i)$$

is exact for all polynomials p of degree $\leq t$. See [6] for an overview of known spherical t-designs for moderate values of t and [3] for a connection between spherical t-designs and group theory.

In [2] it was shown that $N = (t+1)^2$ points $\boldsymbol{x}_i$ are a spherical t-design if $\boldsymbol{c}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \boldsymbol{0}$, where the components of the function $\boldsymbol{c} : \mathbb{R}^{2N-3} \to \mathbb{R}^{N-1}$, $c_i = G_1 - G_{i+1}$, are combinations of the rows of a matrix $G \in \mathbb{R}^{N \times N}$. (Each point is typically given by its polar angles $\varphi \in [0, 2\pi]$, $\theta \in [0, \pi]$, and for symmetry reasons three of these angles can be fixed to $\varphi_1 = \varphi_2 = 0$, $\theta_1 = \pi$. Thus $\boldsymbol{c}$ depends on $2N-3$ scalar arguments.) The entries of the matrix G are defined as

$$G_{ij} = J_t(\boldsymbol{x}_i^T \cdot \boldsymbol{x}_j),$$

with the degree-t polynomial J_t given by

$$J_t(u) = \sum_{i=0}^{t} (2i+1) L_i(u), \tag{1}$$

where the L_i are the Legendre polynomials

$$\left.\begin{array}{rcl} L_0(u) & = & 1, \\ L_1(u) & = & u, \\ L_i(u) & = & \frac{2i-1}{i} u \cdot L_{i-1}(u) - \frac{i-1}{i} L_{i-2}(u), \\ & & i \geq 2. \end{array}\right\} \tag{2}$$

In order to prove the existence of a spherical t-design with $N = (t+1)^2$ points, we apply the Krawczyk operator [8] (modified to handle non-square problems) to verify and enclose a zero of $\boldsymbol{c}$. This modified Krawczyk operator essentially takes the form

$$K([\boldsymbol{z}], \widetilde{\boldsymbol{z}}) = \widetilde{\boldsymbol{z}} - R \cdot \widetilde{\boldsymbol{c}}(\widetilde{\boldsymbol{z}}) + (I - R[C']) \cdot ([\boldsymbol{z}] - \widetilde{\boldsymbol{z}}), \tag{3}$$

where $\widetilde{\boldsymbol{c}}(\boldsymbol{z})$ is a restriction of the function $\boldsymbol{c}$ to a suitable subset of $N-1$ arguments, $[\boldsymbol{z}]$ is an $(N-1)$-dimensional interval vector, $\widetilde{\boldsymbol{z}}$ is some point in $[\boldsymbol{z}]$, R is a nonsingular $(N-1)$-by-$(N-1)$ matrix, and $[C']$ is an enclosure of the derivative of $\widetilde{\boldsymbol{c}}$ over $[\boldsymbol{z}]$; for more details see [1]. The existence of a zero of $\boldsymbol{c}$ (i.e., of an N-point spherical t-design) is guaranteed if

$$K([\boldsymbol{z}], \widetilde{\boldsymbol{z}}) \subset [\boldsymbol{z}] \tag{4}$$

for some $[\boldsymbol{z}]$.

Since each Krawczyk step involves in particular the evaluation of the polynomial J_t at a large number of arguments, the evaluation must take the following observations into account.

- The number of arguments can be very large: Even if we make full use of symmetry, J_t must be evaluated at

$n = N(N+1)/2$ arguments, where $N = (t+1)^2$. Considering the example $t = 40$ ($t = 80$), J_t is evaluated at more than one million (twenty-one million) arguments.

Therefore, the multi-argument evaluation of the polynomial must be fast, and in particular the time-consuming switching of the rounding mode (needed in machine interval arithmetic) should be reduced to a minimum.

- For verified results, the function value $\widetilde{\boldsymbol{c}}(\widetilde{\boldsymbol{z}})$ in Eqn. (3) also must be enclosed in an interval, and therefore Eqn. (4) cannot hold when the diameter of that enclosure significantly exceeds the diameter of $[\boldsymbol{z}]$, which typically is of the order 10^{-8}.

  Thus the evaluation of the polynomial must provide "reasonably sharp" enclosures for the values $J_t(\boldsymbol{x_i}^T \cdot \boldsymbol{x}_j)$.

Note that the argument $u = \boldsymbol{x_i}^T \cdot \boldsymbol{x}_j \in [-1, 1]$ is the cosine of the angle between the two points on the sphere. As can be seen in Fig. 1 for the case $t = 40$, the polynomial J_{40} takes moderate values in the interior of the argument range $[-1, 1]$ and grows for u approaching $+1$, reaching $J_{40}(+1) = 1\,681$. The growth is even stronger for the derivative of J_t, reaching $J'_{40}(1) = 706\,020$.

For any pair of points, the value $u = \boldsymbol{x_i}^T \cdot \boldsymbol{x}_j$ must be computed from the polar angles φ_i, θ_i, φ_j, and θ_j. Due to speed constraints this computation cannot be done with 1 ulp precision. Rather, the argument $[u]$ may have a width up to $\approx 10\varepsilon$, where $\varepsilon \approx 2.2 \cdot 10^{-16}$ denotes the machine precision. Thus we cannot expect the enclosure for $J_t(u)$ to have a diameter smaller than $|J'_t(u)| \cdot \operatorname{diam}[u]$, which can be as large as $706\,020 \cdot 10\varepsilon \approx 1.5 \cdot 10^{-9}$ for $t = 40$.

From Fig. 2 we see that some of the coefficients in the standard representation $J_t(u) = \sum j_k u^k$ are very large, indicating severe cancellation. Thus the evaluation of J_t requires special care even in the interior of the interval, where moderate values for $J'_t(u)$ indicate reasonable conditioning of the problem.

These problems become even more pronounced with increasing t. For $t = 80$, the largest values of the function and the derivative are $J_{80}(+1) = 6\,561$ and $J'_{80}(+1) \approx 10\,760\,040$, and the coefficients of J_{80} do exceed 10^{30}.

2. Methods for evaluating J_t

In this section we discuss several methods for evaluating a polynomial J_t simultaneously at a large number of (interval) arguments. We assess the behavior of these methods with respect to the width of the resulting enclosures, as well as to time.

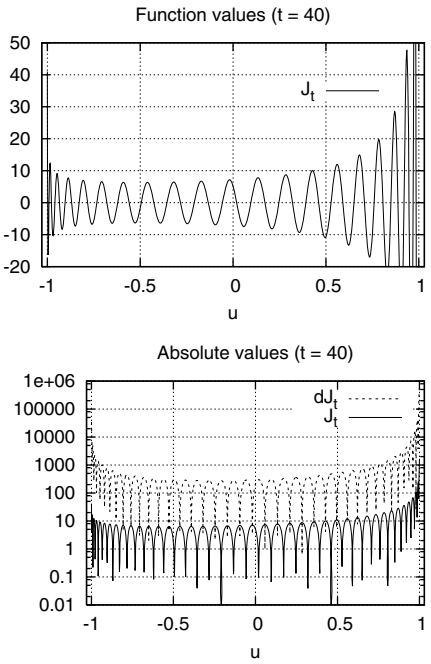


Figure 1. The polynomial J_{40} and its derivative J'_{40} over the relevant range of arguments.

All runs were made on a 2.8 GHz Pentium4 Linux PC with 1 GB of main memory. Since the verification of spherical designs relies heavily on linear algebra computations (both floating-point and interval), the INTLAB [11] environment, which is built on top of MATLAB, was the natural choice for the interval operations. In our experiments, we used version 5.2 of INTLAB (with "fast" interval matrix operations) and version 7.1.0.183 of MATLAB.

The INTLAB operators for interval vectors and matrices employ a minimum number of switchings of the rounding modes. Therefore, interval operations in INTLAB are *very* efficient *if* they can be cast in terms of vectors or matrices. By contrast, frequent switchings of the rounding mode, together with the MATLAB overhead for interpreting instructions, puts a severe penalty on operations with (floating-point or interval) scalars. Note that other interval libraries also provide optimized vector and matrix operations, although the benefits of using them may be less pronounced than in INTLAB.

2.1. The recursive formulation

The first way for evaluating J_t is based on the defining equations (1) and (2).

Note that both equations are amenable to multi-argument evaluation by replacing $[u]$ with a vector or a matrix. Then **1**

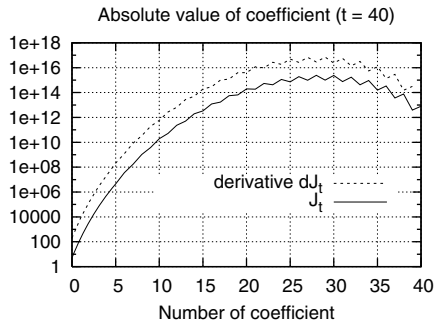


Figure 2. Size of the coefficients of J_{40} and the derivative J'_{40}.

stands for an object of the same shape as $[\boldsymbol{u}]$ with all ones, and the multiplication $[\boldsymbol{u}] \cdot \boldsymbol{L}_{i-1}([\boldsymbol{u}])$ is to be understood entry-wise. In our application, $[\boldsymbol{u}]$ can be taken to be the N-by-N matrix whose (i, j) entry is an enclosure for ${\boldsymbol{x_i}}^T \cdot \boldsymbol{x}_j$.

A straight-forward implementation interleaving (2) with the summation in (1) computes the quantities in the order $\boldsymbol{L}_0$, $\boldsymbol{J}_0$, $\boldsymbol{L}_1$, $\boldsymbol{J}_1$, …, $\boldsymbol{L}_t$, $\boldsymbol{J}_t$ and requires four additional variables $[\boldsymbol{L}_{i-2}]$, $[\boldsymbol{L}_{i-1}]$, $[\boldsymbol{L}_i]$, and $[\boldsymbol{J}]$, each having the shape of $[\boldsymbol{u}]$.

Evaluating the polynomial at all arguments simultaneously makes this method fast, but the results suffer from severe over-estimation, as can be seen in Fig. 3. For $t \geq 20$ this approach is not viable to achieve (4).

2.2. Expanding J_t in the monomial base

By virtue of (1) and (2), the coefficients of the standard representation

$$J_t(u) \quad = \quad \sum_{k=0}^{t} j_k \, u^k$$

can be computed as

$$j_k \quad = \quad \sum_{i=k}^{t} (2i+1)\ell_{i,k}, \quad 0 \leq k \leq t,$$

where

$$\begin{aligned}
\ell_{0,0} &= 1, \\
\ell_{1,0} &= 0, \\
\ell_{1,1} &= 1, \\
\ell_{i,0} &= \frac{1-i}{i}\ell_{i-2,0}, \quad i \geq 2, \\
\ell_{i,k} &= \frac{2i-1}{i}\ell_{i-1,k-1} + \frac{1-i}{i}\ell_{i-2,k}, \quad i \geq 2, k > 0.
\end{aligned}$$

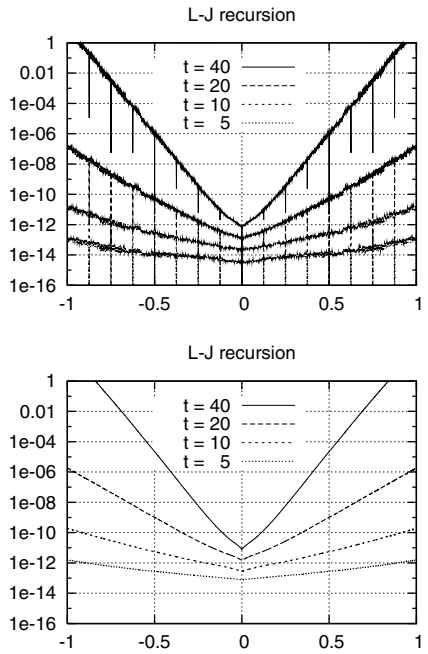


Figure 3. Width of the enclosures for $J_t([u])$ for point arguments $[u] = u \in [-1, 1]$ (upper picture) and for interval arguments with $\operatorname{diam}[u] = 10\varepsilon$ (lower picture), obtained with the method discussed in Sect. 2.1. ε denotes the machine precision, $\varepsilon \approx 2.2 \cdot 10^{-16}$.

These computations are done using *rational arithmetic*, and the resulting coefficients j_k are then enclosed in tight intervals $[j_k]$. Subsequently, a simple Horner scheme

$$\begin{aligned}
&[\boldsymbol{J}] := [j_t] \cdot \mathbf{1} \\
&\textbf{for } k = t-1 : -1 : 0 \\
&\qquad [\boldsymbol{J}] := [\boldsymbol{J}] \cdot [\boldsymbol{u}] + [j_k] \cdot \mathbf{1}
\end{aligned}$$

may be used to evaluate the function. Here, only one additional object $[\boldsymbol{J}]$, which has the shape of $[\boldsymbol{u}]$, and a $(t+1)$-element interval vector $[\boldsymbol{j}]$ are required.

Due to the reduced number of operations this approach is roughly twice as fast as the recursive formulation. However, the widths of the resulting enclosures $J_t([u])$ are almost identical to those shown in Fig. 3, except for the missing spikes at multiples of powers of 2 in the point case, and therefore this method is impractical as well.

2.3. Chebyshev representation

The Legendre polynomials can be represented with the Chebyshev polynomials [4] T_k, which obey the well-known

three-term recurrence relation

$$\left.\begin{array}{rcl} T_0(u) &=& 1, \\ T_1(u) &=& x, \\ T_k(u) &=& 2u \cdot T_{k-1}(u) - T_{k-2}(u),\ k \geq 2. \end{array}\right\} \quad (5)$$

The conversion from Legendre to Chebyshev polynomials is given by

$$\begin{aligned} L_k(u) &= 2g_0 g_k T_k(u) + 2g_1 g_{k-1} T_{k-2}(u) + \ldots \\ &\quad + \left\{ \begin{array}{ll} 2g_{(k-1)/2} g_{(k+1)/2} T_1(u), & k \text{ odd}, \\ g_{k/2}^2, & k \text{ even} \end{array} \right. \end{aligned}$$

with

$$\begin{aligned} g_0 &= 1, \\ g_\ell &= \frac{1 \cdot 3 \cdots (2\ell - 1)}{2 \cdot 4 \cdots (2\ell)}, \quad \ell \geq 1. \end{aligned}$$

Thus

$$J_t(u) = \sum_{k=0}^{t} G_k T_k(u), \quad (6)$$

where

$$G_k = \sum_{i=0}^{t} (2i+1)\gamma_{i,k}$$

with

$$\begin{aligned} \gamma_{k,k-2i} &= 2g_i g_{k-i}, \quad 0 \leq k \leq t, 0 \leq i < k/2, \\ \gamma_{k,0} &= g_{k/2}^2, \quad k \text{ even}. \end{aligned}$$

Again, the coefficients G_k are computed with rational arithmetic and enclosed in tight intervals $[G_k]$, and then the function is evaluated in a straight-forward manner by interleaving (5) with the summation in (6). This implementation requires four $[\boldsymbol{u}]$-shaped objects and a $(t+1)$-element interval vector $[\boldsymbol{G}]$.

However, the widths of the resulting enclosures $J_t([u])$ are again almost identical to those shown in Fig. 3.

2.4. Alternative evaluation of the Chebyshev polynomials

In addition to the standard recurrence (5), the Chebyshev polynomials also obey the log-depth recurrence

$$\left.\begin{array}{rcl} T_0(u) &=& 1, \\ T_1(u) &=& u, \\ T_{2k}(u) &=& 2T_k(u)^2 - 1,\ k \geq 1, \\ T_{2k+1}(u) &=& 2T_k(u) \cdot T_{k+1}(u) - T_1(u),\ k \geq 1. \end{array}\right\} \quad (7)$$

Using this recurrence, J_t is computed according to (6) subsequently to (7).

The log-depth recurrence is one means to achieve the tight enclosures that are necessary for fulfilling Eqn. (4), see Fig. 4. Evaluating the Chebyshev polynomials with this recurrence requires storing most of the $[\boldsymbol{T}_k]$, i.e., $\mathcal{O}(t)$ objects of $[\boldsymbol{u}]$'s shape. If $[\boldsymbol{u}]$ contains millions of interval arguments then the memory requirements may become prohibitive.

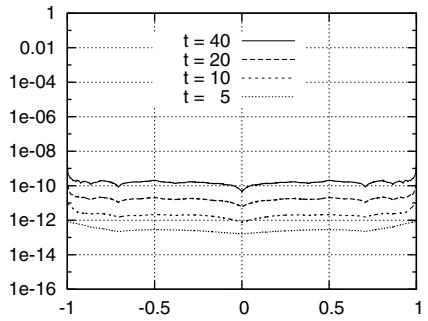


Figure 4. Width of the enclosures for $J_t([u])$, computed with the log-depth Chebychev recurrence, for interval arguments with $\operatorname{diam}[u] = 10\varepsilon$.

2.5. Parameter-dependent linear system

In order to have only machine-representable coefficients, we rewrite the fundamental recursion Eqn. (2) as

$$\begin{aligned} iL_i(u) &= (2i-1)uL_{i-1}(u) - (i-1)L_{i-2}, \\ &\qquad i = 2, 3, \ldots, t. \end{aligned}$$

The vector $\boldsymbol{L}(u) = (L_0(u), \ldots, L_t(u))^T$ thus satisfies the linear system

$$(A_0 + u \cdot A_1)\boldsymbol{L}(u) = (1, 0, \ldots, 0)^T \ (=: \boldsymbol{e}_1 \in \mathbb{R}^{t+1}), \quad (8)$$

where A_0 and A_1 together have just three non-zero diagonals,

$$A_0 = \begin{pmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 1 & 0 & 2 & & & & \\ & 2 & 0 & 3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & t-1 & 0 & t \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0 & & & & & & \\ -1 & 0 & & & & & \\ & -3 & 0 & & & & \\ & & -5 & 0 & & & \\ & & & \ddots & \ddots & & \\ & & & & -(2t-1) & 0 \end{pmatrix}.$$

According to Eqn. (8), the range of $\boldsymbol{L}(u)$ for u varying in some interval $[u]$ corresponds to the parametrized solution set $\Sigma^p := \{\boldsymbol{\ell} \in \mathbb{R}^{t+1} : \exists u \in [u] : (A_0 + uA_1) \cdot \boldsymbol{\ell} = \boldsymbol{e}_1\}$. This set is usually much smaller than the ordinary interval solution set $\Sigma := \{\boldsymbol{L} \in \mathbb{R}^{t+1} : \exists A \in A_0 + [u]A_1 : A \cdot \boldsymbol{L} = \boldsymbol{e}_1\}$. In [7, 9], based on the methods introduced in [10], special techniques have been developed to compute an interval enclosure for Σ^p: If for some matrix $R \in \mathbb{R}^{(t+1)\times(t+1)}$ (usually a computed inverse of $\mathrm{mid}(A_0 + [u]A_1)$), some vector $\hat{\boldsymbol{L}} \in \mathbb{R}^{t+1}$ (an approximate solution to $\mathrm{mid}(A_0 + [u]A_1)\boldsymbol{L} = \boldsymbol{e}_1$, obtained for example by floating-point evaluation of Eqn. (2)) and some interval vector $[\boldsymbol{y}]$ (typically obtained using some steps of ϵ-inflation) one has

$$R \cdot (\boldsymbol{e}_1 - A_0\hat{\boldsymbol{L}}) - [u] \cdot (R \cdot (A_1\hat{\boldsymbol{L}})) + (I - R \cdot A_0 - [u] \cdot (R \cdot A_1)) \cdot [\boldsymbol{y}] \subseteq [\boldsymbol{y}], \quad (9)$$

then $\Sigma^p \subseteq \hat{\boldsymbol{L}} + [\boldsymbol{y}] =: [\boldsymbol{L}]$. We could then obtain an interval enclosing the range of $J_t(u)$ for $u \in [u]$ by evaluating $\sum_{i=0}^{t}(2i+1)[L_i]$. This will, however, still suffer from severe over estimation, since the interval arithmetic evaluation of the sum ignores the fact that all components $[L_i]$ enclose ranges of functions depending on the same parameter.

It is thus appropriate to also include the summation from Eqn. (1) in the formulation of a parametrized linear system, now for $\boldsymbol{J}(u) = (J_0(u), \ldots, J_t(u))^T$. We have

$$B \cdot \boldsymbol{J}(u) = D \cdot \boldsymbol{L}(u),$$

where

$$B = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix},$$

$$D = \mathrm{diag}(1, 3, 5, \ldots, 2t+1).$$

Inserting into Eqn. (8) yields

$$((A_0 + uA_1)D^{-1}B) \cdot \boldsymbol{J}(u) = \boldsymbol{e}_1. \quad (10)$$

Not all entries of the diagonal matrix D^{-1} are exactly representable in floating-point. In order to avoid roundings, we multiply Eqn. (10) from the left with the diagonal matrix $\tilde{D}$ with

$$\tilde{D} = D \cdot \mathrm{diag}(1, 1, 1, 3, 5, \ldots, 2t-3),$$

which turns Eqn. (10) into the system

$$(\tilde{A}_0 + u \cdot \tilde{A}_1) \cdot \boldsymbol{J}(u) = \boldsymbol{e}_1,$$

where the matrices

$$\tilde{A}_0 = \tilde{D}A_0D^{-1}B, \tilde{A}_1 = \tilde{D}A_1D^{-1}B$$

have integer coefficients only. For u varying in $[u]$, we compute an enclosure $[\boldsymbol{J}]$ of its parametrized solution set $\tilde{\Sigma}^p = \{\boldsymbol{J} \in \mathbb{R}^{t+1} : \exists u \in [u] : (\tilde{A}_0 + u\tilde{A}_1) \cdot \boldsymbol{J} = \boldsymbol{e}_1\}$ by testing Eqn. (9) for the new system. We took R as a floating-point inverse of $\mathrm{mid}(\tilde{A}_0 + [u]\tilde{A}_1)$, the approximate solution $\hat{\boldsymbol{J}}$ was obtained by evaluating the recursion Eqn. (2) for $\mathrm{mid}[u]$ in floating-point with subsequent floating-point summation according to Eqn. (1). The ϵ-inflation was started with an initial interval vector $[\boldsymbol{y}]$ centered at zero and with radius vector $|\hat{\boldsymbol{J}}| + \eta$, $\eta = 10^{-15}$. If not successful, the test was repeated with the radius doubled until the test succeeded for the first time.

As can be seen in Fig. 5, this method can yield even tighter enclosures than the log-depth Chebyshev described in Sect. 2.4. But in contrast to all the methods discussed so far, the formulation as a parameter-dependent linear system cannot be implemented in such a way that all evaluations $J_t(u)$ can be done simultaneously. Therefore the rounding mode must be switched much more often than in the other approaches, and, in addition, the total number of operations increases also. Due to these two effects this method is *much* slower than the other ones.

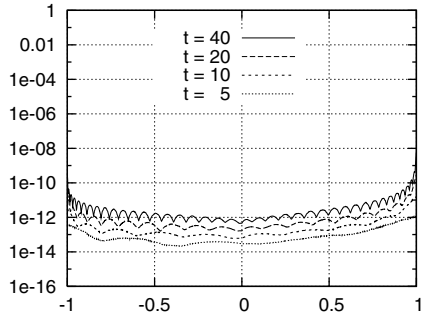


Figure 5. Width of the enclosures for $J_t([u])$, computed with parameter-dependent linear systems, for interval arguments with $\mathrm{diam}\,[u] = 10\varepsilon$.

2.6. Multi-point Horner scheme

This method is based on the observation that even the simple Horner scheme can provide a very sharp enclosure if the argument is small; cf. Fig. 3.

In a preprocessing step we fix $m+1$ points $-1 = \xi_0 < \xi_1 < \ldots < \xi_m = 1$ and determine the expansions of J_t

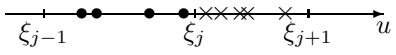


Figure 6. The sets U_j (denoted by •'s) and U_{j+1} (×'s).

around these points,

$$J_t(u) = \underbrace{\sum_{k=0}^{t} j_{k,j}(u-\xi_j)^k}_{=:\, J_{t,\xi_j}(u)}, \quad j = 0, \ldots, m. \tag{11}$$

To obtain the coefficients $j_{k,j}$, we apply the complete Horner scheme in rational arithmetic and enclose the resulting fractions in tight intervals $[j_{k,j}]$. For practical reasons, we take $m = 2^d$ and equidistant points $\xi_j = -1 + j \cdot 2^{1-d}$, $j = 0, \ldots, 2^d$.

To evaluate J_t at a length-n vector $[\boldsymbol{u}]$ containing multiple arguments, we

1. determine a permutation π that sorts the n midpoints of $[\boldsymbol{u}]$ in increasing order,

2. for each $j = 1, \ldots, m$, determine the set U_j of those argument intervals having their midpoint in $[\xi_{j-1}, \xi_j]$; cf. Fig. 6,

3. for each $j = 1, \ldots, m$, do multi-argument evaluations of both expansions $J_{t,\xi_{j-1}}$ and J_{t,ξ_j} for all arguments in U_j (e.g., with Horner's scheme) and intersect the two resulting enclosures.

Since each argument $[u]$ requires the evaluation of two degree-t polynomials, the cost for this approach is approximately twice that of plain Horner evaluation, plus the administrative overhead for preprocessing and for partitioning of the arguments into the sets U_j. Note that the preprocessing is done only *once*. It takes less than 1 second (5 seconds) for $t = 40$ ($t = 80$) and $m = 512$ using e.g., the GNU Multiprecision Arithmetic Library [5], and therefore contributes only a small fraction to the overall time. The partitioning of the n arguments into the m sets takes $\mathcal{O}(n(\log n + \log m))$ operations, if done as described above. There are other partitioning methods not involving an initial sort and requiring a total of $\mathcal{O}(n \log m)$ operations, but sorting and counting seems to be the most efficient technique in INTLAB.

Except for the preprocessing and the partitioning of the arguments, the evaluation time is almost independent from the number of expansion points, $m + 1$. Since there are $2m$ multi-argument Horner evaluations, the rounding mode must be switched more often if m increases. Therefore m should not be chosen too large. In our experiments, we used $m = 512$.

Figure 7 shows that the multi-point Horner scheme can provide very tight enclosures over the whole range of arguments.

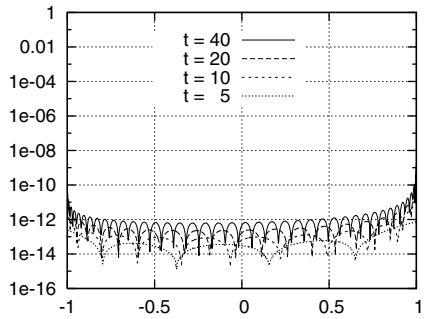


Figure 7. Width of the enclosures for $J_t([u])$, computed with a 513-point Horner scheme, for interval arguments with $\operatorname{diam}[u] = 10\varepsilon$.

2.7. Truncated multi-point Horner scheme

The time for the multi-point Horner scheme can be reduced further by making use of the fact that the higher-order terms $j_{k,j}(u - \xi_j)^k$ contribute only marginally to the sum $J_{t,\xi_j}(u)$ if the argument u is close to the expansion point ξ_j.

To this end, given an (interval) argument vector $[\boldsymbol{u}]$, we determine the *effective degree* ∂_j of the expansion J_{t,ξ_j} by

$$\partial_j = \min\left\{ i : \sum_{k=i+1}^{t} j_{k,j}[\Delta_j]^k \subseteq [-\varepsilon, \varepsilon] \right\}, \tag{12}$$

where $[\Delta_j] = [\underline{u} - \xi_j, \overline{u} - \xi_j]$ and $\underline{u}$ ($\overline{u}$, resp.) denotes the smallest lower (largest upper) bound of any component interval in the vector $[\boldsymbol{u}]$. If the input intervals are very narrow then $\underline{u} \gtrapprox \xi_{j-1} - \xi_j$ and $\overline{u} \lessapprox \xi_{j+1} - \xi_j$ since J_{t,ξ_j} is evaluated only at arguments from U_j and U_{j+1}.

By virtue of Eqn. (12), we have

$$\sum_{k=i+1}^{t} j_{k,j}([u] - \xi_j)^k \subseteq [-\varepsilon, \varepsilon]$$

for each of the arguments $[u]$ in U_j and U_{j+1}, and therefore we can replace the multi-argument evaluation (11) with

$$J_t([\boldsymbol{u}]) \subseteq \sum_{k=0}^{\partial_j} j_{k,j}([\boldsymbol{u}] - \xi_j)^k + [-\varepsilon, \varepsilon]$$

for these arguments. This *truncated* evaluation yields almost exactly the same results as those shown in Fig. 7, but

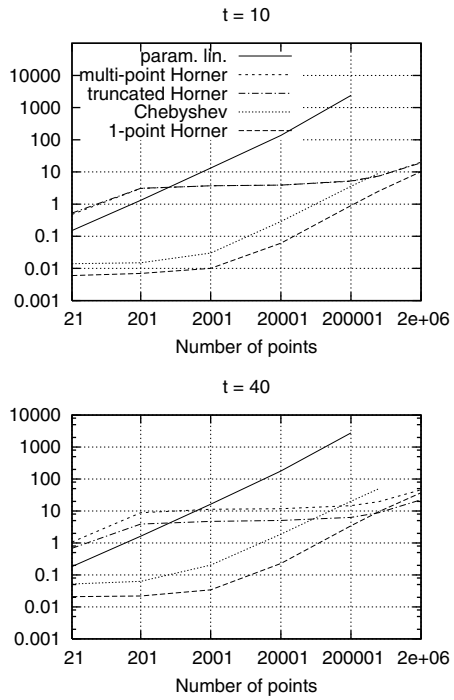


Figure 8. Times (in seconds) for evaluating the polynomials J_{10} (upper picture) and J_{40} (lower picture) with the methods discussed in Sects. 2.1–2.7 at varying numbers of arguments.

since the effective degree ∂_j can be significantly lower than t, the enclosures are obtained faster. To give an example, $t = 40$ and $m = 512$ led to an average effective degree $\partial \approx 11.4$. Increasing the number of expansion points, m, will in general reduce the effective degree, at the cost of increased work for preprocessing and partitioning.

2.8. Timing data

Figure 8 gives the times taken by the methods discussed above to evaluate the polynomials J_{10} and J_{40} at interval vectors with lengths n ranging from 21 to 2 000 001. The Chebyshev evaluation using the log-depth recursion (Sect. 2.4) had to be restricted to vector lengths $\leq 500\,000$ due to memory limitations; the method based on parameter-dependent linear systems (Sect. 2.5) was applied to shorter vectors only to avoid excessive run-times.

At small-to-medium vector lengths, alternative Chebyshev evaluation is the fastest of the viable methods. For $t = 40$, it is roughly eight times slower than plain Horner evaluation, which cannot be used due to extreme over-estimation. For large vector lengths, (truncated) multi-point evaluation (Sects. 2.6 and 2.7) becomes the method of choice, and for $t = 40$ and $n \gtrapprox 10^6$ the truncated method, due to an average effective degree $\partial < 12$, outperforms the Horner evaluation even with respect to time.

Except for very small and very large values of the vector length n, the time for the (truncated) multi-point evaluation is almost constant. It mainly reflects the overhead for partitioning the arguments into the sets U_j and the number of switchings of the rounding mode (this number depends only on m, unless n is very small and most of the U_j are empty, which means that the corresponding Horner evaluations can be skipped completely). By contrast, the cost for arithmetic operations is proportional to n. Therefore arithmetic operations dominate the overall time for very long vectors.

3. Application of the methods in similar contexts

The methods discussed in the preceding section also can be applied in slightly different settings, as explained in the following.

3.1. Evaluating derivatives on wider intervals

Computing the matrix $[C']$ in Eqn. (3) also requires an evaluation of the derivative $J_t'([\boldsymbol{z}])$. This can be done with each of the techniques discussed in Sect. 2.

It should be noted that the diameter of the arguments $[z]$ in the derivative evaluation is considerably larger than the diameter of the arguments in the evaluation of the function J_t. Typical values are $\mathrm{diam}[z] \sim 10^{-8}$, whereas in the evaluation of J_t we have $\mathrm{diam}[u] \approx 10\varepsilon \sim 10^{-15}$. For all the methods, increasing the width of the arguments leads to a roughly proportional growth of the enclosures for J_t; see Fig. 9, which clearly shows the 10^7-fold increase of $\mathrm{diam}(J_t)$ when $\mathrm{diam}[u]$ is increased by the same factor, as compared to Fig. 7. Note that the plots in Fig. 9 are also for J_t to facilitate the comparison. The width of the enclosure for the *derivative* is larger by another factor of ≈ 500 (for $t = 40$).

In our application, even $\mathrm{diam}[C'] \sim 1$ is no serious obstacle to achieving (4) since the matrix $I - R \cdot [C']$ in Eqn. (3) has small entries (R is chosen to approximate the inverse of $\mathrm{mid}[C']$), and thus the matrix–vector multiplication reduces the width.

3.2. Multi-argument evaluation of general polynomials

The truncated multi-point Horner evaluation adapts in a straight-forward manner to the evaluation of general polynomials p at a large number of arguments, provided that

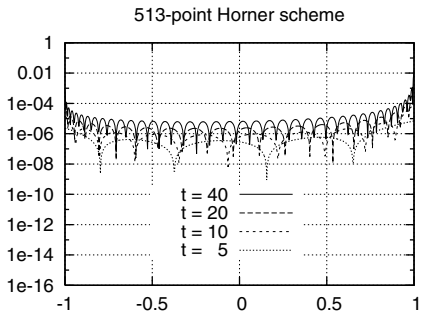


Figure 9. Width of the enclosures for $J_t([u])$, computed with multi-point Horner evaluation, for interval arguments with $\operatorname{diam}[u] = 10^8\varepsilon$.

1. the coefficients γ_k in the monomial expansion $p(u) = \sum \gamma_k u^k$ are given or can be computed *exactly*, and
2. the arguments u are contained in a given compact interval $[\alpha, \beta]$.

To avoid unnecessary rounding errors, the expansion points ξ_j should be machine numbers. In fact they should be representable with a short mantissa in order to avoid excessive growth of the numerators and denominators in the coefficients $\gamma_{k,j}$ of the polynomials $p_{\xi_j}(\cdot) = p(\cdot - \xi_j)$.

Note that the expansion points ξ_j need not be equidistant. For example, an adaptive subdivision strategy might be used where the distance of the points is controlled via the "local" effective degree, which in turn depends on the higher derivatives of p. To give an example, 65 out of the 513 expansion points (with increasing distances toward the midpoint of the interval $[-1, 1]$) are sufficient to yield enclosures for J_{40} that are almost identical to those reported in Fig. 7.

If there are no a priori bounds on the arguments then α and β, and thus the expansion points ξ_j and the coefficients $\gamma_{k,j}$, may be determined within the multi-argument evaluation. This just adds the overhead for preprocessing to each evaluation call.

By contrast, if only enclosures for the coefficients γ_k are available then this approach in general will lead to severe over-estimation already in the computed $\gamma_{k,j}$, and therefore in the final enclosures as well.

4. Concluding remarks

We have presented a truncated multi-point Horner scheme for evaluating a certain degree-t polynomial J_t at a large number of (interval) arguments. This method relies on computing the expansions of the polynomial around several points and then truncating these expansions to an appropriate "effective degree". Our experiments showed that our approach is very efficient (indeed, it can be faster than using the standard Horner scheme) and yields very tight enclosures for the function values. The method is easily adapted to the multi-argument evaluation of other polynomials given in the standard monomial representation.

Using the truncated multi-point Horner scheme, we were able to prove the existence of $(t+1)^2$-point spherical designs for t values up to $t = 80$ [1]. This is a major improvement as compared with the previous limitation to $t \leq 20$ [2], given that the evaluation of the function $\boldsymbol{c}$ alone scales with the fifth power of t.

Acknowledgements

The authors would like to thank Knut Petras for pointing them to the log-depth recursion for the Chebyshev polynomials.

References

[1] X. Chen, A. Frommer, and B. Lang. Fast and rigorous verification of shperical t-designs, 2006. In Preparation.

[2] X. Chen and R. S. Womersley. Existence of solutions to systems of underdetermined equations and spherical designs. *SIAM J. Matrix Anal. Appl.*, 2006. To appear.

[3] P. de la Harpe and C. Pache. Spherical designs and finite group representations (some results of E. Bannai). *European J. Combin.*, 25(2):213–227, 2004.

[4] L. Fox and I. B. Parker. *Chebyshev Polynomials in Numerical Analysis*. Oxford University Press, London, UK, 1968.

[5] GNU MP—the GNU multiprecision arithmetic library, May 2006. http://www.swox.com/gmp/manual/.

[6] R. H. Hardin and N. J. A. Sloane. McLaren's improved snub cube and other new spherical designs in three dimensions. *Discrete Comput. Geom.*, 15:429–441, 1996. Updated 2002, http://www.research.att.com/~njas/doc/snub.ps.

[7] W. Krämer and E. D. Popova. Zur Berechnung von verlässlichen Außen- und Inneneinschließungen bei parameterabhängigen linearen Gleichungssystemen. *Proc. Appl. Math. Mech.*, 4:670–671, 2004.

[8] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.

[9] E. D. Popova. Parametric interval linear solver. *Numer. Algorithms*, 37(1–4):345–356, 2004.

[10] S. M. Rump. Verification methods for dense and sparse systems of equations. In J. Herzberger, editor, *Topics in Validated Computations*, pages 63–136. Elsevier North-Holland, Amsterdam, 1994.

[11] S. M. Rump. INTLAB—INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

Vectorised/Semi-Parallel Interval Multiplication

Eoin Malins, Marek Szularz, Bryan Scotney
University of Ulster at Coleraine, Coleraine, Northern Ireland, UK
eoin@infc.ulst.ac.uk, marek@infc.ulst.ac.uk, bw.scotney@ulster.ac.uk

Abstract

To date, two principle methods for the multiplication of two intervals have been proposed. Namely, the multiplication of all four bounds and finding their minima/maxima; or by pre-processing the bounds and determining which multiplicands to use based upon their signs. In either case, a minimum of four multiplications are required for complete coverage and special cases such as [0, 1] × [-∞, 1] can result in the less than enlightening [-∞, +∞]. This paper describes a new method of interval multiplication that never requires more than two multiplications, has no special cases and elegantly handles the above case. We continue by describing reformulations of the brute-force and 9-case methods which, through making use of SIMD technology, parallelise and vectorise their operation, ultimately allowing the complete removal of branching. We conclude with an analysis of the algorithms and their performance, compared with the two forementioned traditional techniques.

1. Introduction

In an ideal world, the product of two intervals would require only two multiplications, those of the bounds guaranteed to produce the upper and lower endpoints. To date, several techniques have been proposed (Interval Multiplication, Double-Precision Multiplication and Case-based Multiplication), each requiring 8, 4 or 2.2[1] floating-point multiplications respectively. We review the above methods and propose a new one in which by using integers to pre-process the bounds allows a reduction to this ideal of only two.

There is a growing trend in processor design towards the addition of 'multimedia' (SIMD (Single Instruction Multiple Data)) units, where a single operation is executed in parallel on multiple independent pieces of data. Though the data which may be processed in this manner includes floating-point, sadly such systems do not natively lend themselves to interval operations, as while they are able to perform the same task on multiple data elements, all results must be rounded in the same direction. However, these units can facilitate interval operations by way of restructuring the nature of these operations and the supporting routines, (such as determining the minimum/maximum value) on which they rely.

By use of these techniques and re-examining the condition table for the 9-case method, in the following pages we describe and demonstrate reformulations of the traditional brute-force and 9-case interval multiplication algorithms which are completely vectorised. We conclude with a synopsis of the factors involved in the implementation of said algorithms followed by their relative performances.

2. Brute-Force Interval Multiplication

Several techniques for interval multiplication have been described by Moore in [3]. The simplest approach (shown in 1) dictates that each bound should be multiplied by every other bound; the lower and upper bounds of the resultant interval being (respectively) the minima and maxima of this calculation. In order to compensate for rounding error and thus ensure that the result fully encloses all possible values, the lower bound (denoted by $\underline{x}$) must be rounded towards -∞ (denoted by ∇) whilst the upper bound ($\overline{x}$) is rounded towards +∞ (Δ).

$$\begin{aligned}\mathbf{Z} = \mathbf{X} \times \mathbf{Y} = \quad & [min\{\nabla\underline{xy},\ \nabla\underline{x}\overline{y},\ \nabla\overline{x}\underline{y},\ \nabla\overline{xy}\}, \\ & max\{\Delta\underline{xy},\ \Delta\underline{x}\overline{y},\ \Delta\overline{x}\underline{y},\ \Delta\overline{xy}\}] \quad (1)\end{aligned}$$

This method requires eight floating-point multiplications and six comparisons in order to determine the minima and maxima.

Should a double precision product be available, an improvement on this method, shown by Stine and Schulte in [7] allows the result to be computed as:

[1] This is an average, explained in Section 3.

$$\mathbf{Z} = \mathbf{X} \times \mathbf{Y} = \begin{array}{l}[\nabla min\{\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\}, \\ \Delta max\{\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\}]\end{array} \quad (2)$$

This allows a reduction from 8 multiplications to only 4; whilst the number of comparisons remains the same.

While simple to implement, this system suffers from overheads in that twice as many floating point operations are calculated than ultimately used and due to this, six IF statements must be evaluated to compensate. In an ideal world, an interval multiplication would only require two multiplications.

3. Case Based Multiplication

Implied by Moore in [3] and described by Schulte and Swartzlander in [6], an improvement on the above methods allows in 8 out of 9 cases a further reduction from 4 to only 2 multiplications. This is achieved by examining the signs of the operands prior to multiplication. Given the data shown in Table 1 it is possible to select which multiplicands which will always produce the smallest/largest bounds.

Case	Conditions	$\underline{z}$	$\overline{z}$
1	$\underline{x} \geq 0, \underline{y} \geq 0$	$\underline{x}\underline{y}$	$\overline{x}\overline{y}$
2	$\underline{x} \geq 0, \overline{y} < 0$	$\overline{x}\underline{y}$	$\underline{x}\overline{y}$
3	$\overline{x} < 0, \underline{y} \geq 0$	$\underline{x}\overline{y}$	$\overline{x}\underline{y}$
4	$\overline{x} < 0, \overline{y} < 0$	$\overline{x}\overline{y}$	$\underline{x}\underline{y}$
5	$\underline{x} < 0 \leq \overline{x}, \underline{y} \geq 0$	$\underline{x}\overline{y}$	$\overline{x}\overline{y}$
6	$\underline{x} < 0 \leq \overline{x}, \overline{y} < 0$	$\overline{x}\underline{y}$	$\underline{x}\underline{y}$
7	$\underline{x} \geq 0, \underline{y} < 0 \leq \overline{y}$	$\overline{x}\underline{y}$	$\overline{x}\overline{y}$
8	$\overline{x} < 0, \underline{y} < 0 \leq \overline{y}$	$\underline{x}\overline{y}$	$\underline{x}\underline{y}$
9	$\underline{x} < 0 \leq \overline{x}, \underline{y} < 0 \leq \overline{y}$	*min of* ($\underline{x}\overline{y}$, $\overline{x}\underline{y}$)	*max of* ($\underline{x}\underline{y}$, $\overline{x}\overline{y}$)

Table 1. Nine cases for interval multiplication. [7]

In [5] Schulte, Bickerstaff and Swartzlander point out that although this decreases the number of multiplications, its software implementation requires a large number of conditional branches to determine the sign bits. Specifically, this implementation comes at the cost of if-then-else statements which are nested three deep.

It should be noted that the above system suffers from a 'special case' (case 9) in which the multiplicands cannot be determined in advance and so the system must fall back to multiplying out all possible combinations and subsequently sorting them. Again, should double precision results prove unavailable, all the overheads in Equation (2) will once more be incurred. Namely: 8 multiplication operations, followed by 6 comparison operations.

4. Integer Multiplication

Our technique for interval multiplication involves the matrix multiplication of integers which symbolically represent the interval bounds. As with any symbolic operation, it is not the values which are important but the signs and ratios of these values. At the expense of performing this calculation prior to multiplying the bounds, we can always correctly identify which combination of multiplicands will produce the smallest and largest bounds; thus reducing the number of floating point operations from a worst-case of eight to the ideal of only two.

Whilst admittedly there are overheads incurred in the pre-processing, it should be borne in mind that any of the above techniques use some variety of pre or post processing in order to determine the correct bounds. The primary advantages of this technique are:

- Maximum of two floating-point multiplications per operation
- Fixed number of operations per multiplication
- No special cases
- Low cost hardware implementation

For example, given the following intervals:

$$\mathbf{X} = [-1.32, 102.46] \quad (3)$$

$$\mathbf{Y} = [22.10, 112.41] \quad (4)$$

$$\mathbf{X} \times \mathbf{Y} = [-1.32, 102.46] \times [22.10, 112.41] \quad (5)$$

The integer representation of the absolute values of these would be sorted as follows:

$$[1, 3] \times [2, 4] \quad (6)$$

Which, reapplying their relevant signs would become:

$$[-1, 3] \times [2, 4] \quad (7)$$

Or more generally:

$$[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \quad (8)$$

Calculating the outer product on the above integer values gives:

$$\begin{pmatrix} -1 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 2 & 4 \end{pmatrix} = \begin{pmatrix} -2 & -4 \\ 6 & 12 \end{pmatrix} \quad (9)$$

Which is generalised to the following:

$$\begin{pmatrix} \underline{x} \\ \overline{x} \end{pmatrix} \cdot \begin{pmatrix} \underline{y} & \overline{y} \end{pmatrix} = \begin{pmatrix} \underline{x}\underline{y} & \underline{x}\overline{y} \\ \overline{x}\underline{y} & \overline{x}\overline{y} \end{pmatrix} \quad (10)$$

Thus the minimum (-4) and maximum (12) values refer to $\underline{x}\overline{y}$ and $\overline{x}\overline{y}$, i.e. the interval $[\underline{x}\overline{y}, \overline{x}\overline{y}]$ ([-148.38, 11517.53]) which is the product of $\mathbf{X}\cdot\mathbf{Y}$.

4.1 Operation comparison

Figure 1 shows the only six valid types of interval, their magnitude increasing from left to right with zero lying at any point in the graphs. All of the cases described by the 9-case method in Table 1 fall within one of these types. For example, case 1 from Table 1, is such that $\underline{x} > 0$ and exhibits $\underline{x} < \overline{x}\underline{y} < \overline{y}\overline{x}$ thus conforming to interval type A from Figure 1.

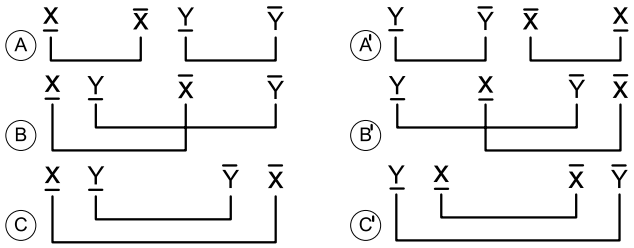


Figure 1. Valid interval types

Exhaustively covering every possible interval which can be processed by the 9-case method, Table 2 takes each case from the 9-case method and determines which of the six interval types the case utilises. An example of input operands for each case and type is given ($\mathbf{X}$ and $\mathbf{Y}$) and the subsequent integer multiplication values calculated using the generalised form given in Equation (10). Taking the minimum and maximum values (imin and imax respectively), the bounds which these integers represent produce a product which in every case is identical to that of the 9-case method.

4.2. Special Cases

According to the interval specification in [1] the calculation:

$$[0, 1] \times [-\infty, 1] = [-\infty, 1] \tag{11}$$

Is defined as:

$$[0, 1] \times [-\infty, 1] = [-\infty, +\infty] \tag{12}$$

This is due to the multiplication of $\pm\infty$ by 0 being undefined and as such, when using floating-point arithmetic, the product is defined as the special value 'NaN' (Not A Number). Subsequent comparisons between NaN and the real values generated from the other three boundary candidates will also result in NaN. Thus, as a NaN does not contain any meaningful boundary data, in order to ensure that all possible values are enclosed the bounds must be extended to $[-\infty, +\infty]$. However, given the limits of interval bound $\boldsymbol{n}$ set out in Equation (13), we can see Equation (11) provides a better result.

$$[0, 1] \times [-\boldsymbol{n}, 1] = [-\boldsymbol{n}, 1] \tag{13}$$

$$\lim_{\boldsymbol{n}\to\infty} [-\infty, 1] \tag{14}$$

An advantage of our technique is its ability to gracefully handle problems such as the above. Using the technique described earlier, we can show:

$$\mathbf{X} = [0,\ 1] \tag{15}$$

$$\mathbf{Y} = [-\infty,\ 1] \tag{16}$$

Would be sorted[2] as:

$$[1,\ 2] \times [3,\ 2] \tag{17}$$

Upon reapplying the signs to become:

$$[1,\ 2] \times [-3,\ 2] \tag{18}$$

Calculating the outer product on the above integer values gives:

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} -3 & 2 \end{pmatrix} = \begin{pmatrix} -3 & 2 \\ -6 & 4 \end{pmatrix} \tag{19}$$

I.e:

$$\begin{pmatrix} \underline{x} \\ \overline{x} \end{pmatrix} \cdot \begin{pmatrix} \underline{y} & \overline{y} \end{pmatrix} = \begin{pmatrix} \underline{x}\underline{y} & \underline{x}\overline{y} \\ \overline{x}\underline{y} & \overline{x}\overline{y} \end{pmatrix} \tag{20}$$

Thus the minimum (-6) and maximum (4) values refer to $\overline{x}\underline{y}$ and $\overline{x}\overline{y}$, which are the correct combination of bounds for this interval multiplication. Thus, multiplying the values these refer to gives the result: [-∞, 1]

5. Vectorisation

A vector may be described as a one-dimensional array, or single path, thus a vectorised program is a single instruction path which executes without branching. As CPU speeds have increased, the time for instructions to be completed has decreased proportionately to the extent where some instructions cannot be executed in the time allotted to them. In an attempt to rectify this, most architectures will cut the instruction into simpler sub-tasks which are faster to process and are executed serially in a pipeline. Whilst for the majority of tasks this is a satisfactory solution, many operations such as a change of rounding direction or code branch (IF statement) force pipeline stalls or even flushes.

[2] As $\overline{x}$ and $\overline{y}$ are the same value, they are both assigned the same integer representation.

Case	Type	**X**	**X**	$\{\underline{xy}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}\}$	imin	imax	integer/9-case result
1	A	[1, 2]	[3, 4]	{3, 4, 6, 8}	3	8	$[\underline{xy},\overline{xy}]$
	B	[1, 3]	[2, 4]	{2, 4, 6, 12}	2	12	$[\underline{xy},\overline{xy}]$
	C	[1, 4]	[2, 3]	{2, 3, 8, 12}	2	12	$[\underline{xy},\overline{xy}]$
	A'	[3, 4]	[1, 2]	{3, 6, 4, 8}	3	8	$[\underline{xy},\overline{xy}]$
	B'	[2, 4]	[1, 3]	{2, 6, 4, 12}	2	12	$[\underline{xy},\overline{xy}]$
	C'	[2, 3]	[1, 4]	{2, 8, 3, 12}	2	12	$[\underline{xy},\overline{xy}]$
2	A'	[1, 2]	[-4, -3]	{-4,-3,-8,-6}	-8	-3	$[\overline{x}\underline{y},\underline{x}\overline{y}]$
3	A	[-2, -1]	[3, 4]	{-6,-8,-3,-4}	-8	-3	$[\underline{x}\overline{y},\overline{x}\underline{y}]$
4	A	[-4, -3]	[-2, -1]	{8, 4, 6, 3}	3	8	$[\overline{xy},\underline{xy}]$
	B	[-4, -2]	[-3, -1]	{12, 4, 6, 2}	2	12	$[\overline{xy},\underline{xy}]$
	C	[-4, -1]	[-3, -2]	{12, 8, 3, 2}	2	12	$[\overline{xy},\underline{xy}]$
	A'	[-2, -1]	[-4, -3]	{8, 6, 4, 3}	3	8	$[\overline{xy},\underline{xy}]$
	B'	[-3, -1]	[-4, -2]	{12, 6, 4, 2}	2	12	$[\overline{xy},\underline{xy}]$
	C'	[-3, -2]	[-4, -1]	{12, 3, 8, 2}	2	12	$[\overline{xy},\underline{xy}]$
5	A	[-1, 2]	[3, 4]	{-3, -4, 6, 8}	-4	8	$[\underline{x}\overline{y}, \overline{xy}]$
	B	[-1, 3]	[2, 4]	{-2, -4, 6, 12}	-4	12	$[\underline{x}\overline{y}, \overline{xy}]$
	C	[-1, 4]	[2, 3]	{-2, -3, 8, 12}	-3	12	$[\underline{x}\overline{y}, \overline{xy}]$
6	C	[-4, 1]	[-3, -2]	{12, 8, -3, -2}	-3	12	$[\overline{x}\underline{y}, \underline{xy}]$
	A'	[-2, 1]	[-4, -3]	{8, 6, -4, -3}	-4	8	$[\overline{x}\underline{y}, \underline{xy}]$
	B'	[-3, 1]	[-4, -2]	{12, 6, -4, -2}	-4	12	$[\overline{x}\underline{y}, \underline{xy}]$
7	A'	[2, 3]	[-4, 1]	{-8, 2, -12, 3}	-12	3	$[\overline{x}\underline{y}, \overline{xy}]$
	B'	[1, 3]	[-4, 2]	{-4, 2, -12, 6}	-12	6	$[\overline{x}\underline{y}, \overline{xy}]$
	C'	[2, 3]	[-1, 4]	{-2, 8, -3, 12}	-3	12	$[\overline{x}\underline{y}, \overline{xy}]$
8	A	[-4, -3]	[-2, 1]	{8, -4, 6, -3}	-4	8	$[\underline{x}\overline{y}, \underline{xy}]$
	B	[-4, -3]	[-2, 1]	{8, -4, 6, -3}	-4	8	$[\underline{x}\overline{y}, \underline{xy}]$
	C'	[-3, -2]	[-4, 1]	{12, -3, 8, -2}	-3	12	$[\underline{x}\overline{y}, \underline{xy}]$
9	B	[-4, 1]/[-2, 3]	[-3, 2]/[-1, 4]	{12, -8, -3, 2}/{2, -8, -3, 12}	-8	12	$[\underline{x}\overline{y}, \underline{xy}]$/ $[\underline{x}\overline{y}, \overline{xy}]$
	C	[-4, 2]/[-2, 4]	[-3, 1]/[-3, 1]	{12, -4, -6, 2}/{6, -2, -12, 4}	-6/-12	12/6	$[\overline{x}\underline{y}, \underline{xy}]$/ $[\overline{x}\underline{y}, \underline{xy}]$
	B'	[-3, 2]/[-1, 4]	[-4, 1]/[-2, 3]	{12, -3, -8, 2}/{2, -3, -8, 12}	-8	12	$[\overline{x}\underline{y}, \underline{xy}]$/ $[\overline{x}\underline{y}, \overline{xy}]$
	C'	[-3, 1]/[-1, 3]	[-4, 2]/[-2, 4]	{12, -6, -4, 2}/{2, -4, -6, 12}	-6	12	$[\underline{x}\overline{y}, \underline{xy}]$/ $[\overline{x}\underline{y}, \overline{xy}]$

Table 2. Valid intervals and products for each of the nine cases, as compared to the results of the integer method.

IF statements cause a particular problem as with so little time for the condition to be evaluated and instructions waiting further up the pipeline, it is undesirable to simply pause execution while the appropriate branch is determined. To this end, a specialist piece of hardware, the 'branch prediction unit' attempts to guess which path the branch will take and begin speculative execution of that branch. Whilst this is sufficient for structures such as for loops, where the loop control variable must be evaluated regularly, and the branch is liable to be the same as the previous iteration, the branch prediction logic can 'learn' from experience. (i.e. if the loop went around the previous 50 times, it is liable to go round on iteration 51 and so this path starts speculative execution). Should the prediction prove incorrect, the pipeline must be flushed and all intermediate results cleared and re-executed with the correct data.

This poses a particular problem for interval arithmetic due to the number of IF statements required and the execution path being data dependent, something which changes with each multiplication. Subsequently the branch predictor cannot learn what to expect and a misprediction results in a pipeline flush, which on a modern architecture such as the Pentium 4 is 40 instructions deep.

Along with deeper pipelines, modern processor design is leaning towards the inclusion of SIMD (Single Instruction Multiple Data) units. The following Sections detail how SIMD data structures can be used to vectorise interval code and remove these costly branching operations.

5.1. Vectorised Brute-Force

The single-precision brute-force algorithm given in Equation (1) makes use of eight floating point multiplications followed by six comparisons in order to determine which products are the minima/maxima for the given pair of intervals.

Current SIMD technology can simultaneously process four single precision floats and so the initial multiplications may be replaced by two SIMD multiplications, provided the rounding direction is set appropriately. However the six comparisons prove a more interesting problem as without IF statements, it is not immediately clear how to return the minimum value in code such as that shown in Figure 2.

```
int min(int a, int b)
{
    if(a < b)
    {
        return a;
    }else{
        return b;
    }
}
```

Figure 2. Function to return the smaller of two ints.

By making use of masking and SIMD operations, it is possible to produce code which performs the same function as that shown in Figure 2 but which does not require IFs and the branches these produce. An example similar to that given in [4] for an Altivec architecture is shown in Figure 3.

```
vector int Min(vector int a, vector int b)
{
    vector bool mask = vec_cmplt( a, b )
    /* Generate mask for smaller value */

    vector int result = vec_sel( a, b, mask )
    /* Select a or b using the mask */

    return result
}
```

Figure 3. Non-branching implementation of Figure 2

The code in Figure 3 takes two vectors containing signed ints and using the instruction `vec_cmplt(2)` returns a bitmask highlighting the smaller value. When passed to `vec_sel(3)` with the input vectors, the bitmask selects and returns the single vector which has been masked off. SIMD capabilities vary between architectures, the Intel architectures for example, do not contain the vec_sel instruction, so it may be more or less difficult to implement these functions depending upon the choice of platform.

The pseudocode in Figure 4 shows an implementation of the brute-force interval multiplication which makes use of relatively simple functions. For the product of two input intervals $\mathbf{X}$ and $\mathbf{Y}$ the resultant upper/lower bound will be generated in the every element of the array.

```
interval X = [1, 3]
/* X = [x, x̄] */
interval Y = [2, 4]
/* Y = [y, ȳ] */

union vector
{
  128 bits v
  float    f[4]
}product,
 dataA, dataB
 rot1, imed_val
 rot2, vec_min

dataA = {x, x, x̄, x̄}
/* dataA =  [1,  1,  3,  3] */
dataB = {y, ȳ, y, ȳ}
/* dataB =  [2,  4,  2,  4] */

rounding_mode = ROUND_DOWN
/* Set rounding mode down */

product = SIMD_mult(dataA, dataB)
/* product = [2,4,6,12] */

rot1 = rotate_right_1(product)
/* rot1 = [12,2,4,6] */

imed_val = cmplt(product, rotr1)
 /*   product  :  2,4,6,12
      rot1     : 12,2,4,  6
      imed_val =  2,2,4,  6
 /*

rot2 = rotate_right_2(imed_val)
 /*   imed_val : 2,2,4,6
                2,2,4,6 >> 6,2,2,4 >> 4,6,2,2
          rot2 = 4,6,2,2
 /*

vec_min = cmplt(imed_val, rot2)
/* vec_min = [2,2,2,2] */
```

Figure 4. Vectorised/semi-parallel implementation of the brute force algorithm.

In this system, a set of unions each contain a 128-bit vector and an array containing four floating point variables. The union allows the vector to be processed in the SIMD XMM registers, whilst still allowing access to its individual components via the array. With the rounding mode set to ROUND_DOWN, two vectors, *dataA* and *dataB* are assembled so their SIMD multiplication via function `SIMD_mult(2)` will result in the vector variable *product* containing products of the input bounds; (i.e. $\underline{x}\underline{y}$, $\underline{x}\overline{y}$, $\overline{x}\underline{y}$

and $\overline{x}\overline{y}$). In order to determine the smallest of the four floats contained within *product*, six comparisons are necessary, though on the Intel architecture, only 4 may be performed at a time. In order to make the comparisons, we must set up a second vector *rot1* containing the same data, but in a different order. To do this, our pseudo-instruction `rotate_right_1(1)` takes *product* and 'rotates' the values it contains by a specified amount and direction to produce the vector *rot1*. For example, rotating the array 1,2,3,4 right by one would result in 4,1,2,3. The instruction `cmplt(2)` (compare less than) then compares the values contained in *product* with those in *rot1* and in each case returns the smaller of these, in the vector *imed_val*. At this point, four of the required six comparisons have been made and to perform the remaining two, a second rotate (this time of *imed_val*) is required. By rotating *imed_val* right (or left) by two elements, the uncompared pairs of elements are now aligned with each other. With a final `cmplt(2)` comparing the elements of *imed_val* and *rot2*, all values have been compared and all elements in *min_val* contain the same (minimum) value. Thus, the lower bound of the output interval can be set to any of the elements of *min_val*. In order to compute the upper bound, the same technique would be followed, though having changed the initial rounding mode to 'ROUND_UP' and using `cmpgt(2)` (compare greater than) instead of `cmplt(2)`. This technique, ensures that no branching is required and eliminates pipeline stalls.

5.2. Vectorised 9-case implementation

The following implementation of the 9-case method describes a means of completely removing branching from the algorithm and minimizing the number of and types of comparison necessary to determine the case to be executed.

While branching is not such an issue for the Brute-Force algorithm, as it takes place after the calculation of the floating-point values, when branching takes place *before* sending data to the FPU, there is a significant overhead. The cost of a misprediction, especially when feeding relatively high-latency units such as floating-point multipliers can result in multiple pipeline stalls or flushes.

Though the Min/Max operations had a series of IF statements removed, vectorising code which makes use of nested IFs is a significantly more complex task. However, given the conditions described in Table 1 we can see there are additional implied conditions which will simplify our task. To take an example from case 1 ($\underline{x} \geq 0$, $\underline{y} \geq 0$); as $\underline{x}$ is greater than zero, the upper bound of this interval ($\overline{x}$) must also be greater than zero. By filling in the implied conditions as well, we gain Tables 3 and 4.

As Tables 3 and 4 demonstrate, each of the conditions distinctly identifies one of the original cases in Table 1. As both tables are independently capable of uniquely identifying each case and Bound ≥ 0, or Bound < 0, are used equally frequently, either may be used. As the same operation is applied to each of the interval bounds, this lends itself to a SIMD implementation. By simultaneously comparing each of the operands to zero, a bit-pattern determining which of the above cases is in operation is produced. Casting this bit pattern to an integer variable produces a value in the range 0 (case 1) to 15 (case 4).

	Bound < 0			
	$\underline{x}$	$\overline{x}$	$\underline{y}$	$\overline{y}$
1	F	F	F	F
2	F	F	T	T
3	T	T	F	F
4	T	T	T	T
5	T	F	F	F
6	T	F	T	T
7	F	F	T	F
8	T	T	T	F
9	T	F	T	F

Table 3. Expanded bound $<$ 0 conditions for 9-case interval multiplication.

	Bound ≥ 0			
	$\underline{x}$	$\overline{x}$	$\underline{y}$	$\overline{y}$
1	T	T	T	T
2	T	T	F	F
3	F	F	T	T
4	F	F	F	F
5	F	T	T	T
6	F	T	F	F
7	T	T	F	T
8	F	F	F	T
9	F	T	F	T

Table 4. Expanded bound $\geq$ 0 conditions for 9-case interval multiplication.

At this point, although we could have a case statement to perform the necessary calculations based upon the case in hand, this would be the equivalent of 9 logical IF statements, something we are trying to avoid. Instead, the integer is used as an index to a 16-element array containing pointers to functions. Each of these functions contains code necessary to calculate the bounds for one case. Thus, the array elements are arranged such that the integer generated indexes the correct function. Though there are only 9 cases (0,3,12,15,8,11,2,14,10), but 15 possible integers; the remaining (invalid) array elements are filled by pointers to a function which returns [NaN, NaN].

It should be noted that normally, function calls generate an additional overhead. The time taken to call a function (setting its arguments, loading cache/memory, jumping to the new instructions, executing them and jumping back) is normally a very small part of the overall execution time. When the called function is relatively simple but used heavily, the function call overhead and the execution time are similar and overheads quickly mount up. In the vectorised 9-case method, though individual cases were generated as separate functions, the structure which indexes them (the array of pointers) is static and so assembles to produce code which makes use of a JMP table, but no function calls and the overhead which these incur.

6. Testing

Each of the above algorithms were required to perform the same task, with the same data: to calculate the product of two random intervals in the range $[-\infty, +\infty]$ 1×10^8 times.

Each routine was implemented as a discrete function, accepting two pointers to intervals as parameters and returning another pointer as the result. A loop would generate two random intervals passing each to the above functions before repeating. As each function made the same calculation with the same values, their individual performances could be determined by the length of time the program spent in each function, as determined by gprof[3]. They were implemented using C90 compiled with GCC v3.2 on an Intel PentiumIII Xeon[4]. In order to ensure that no function would gain from using the rounding mode set by its predecessor, the rounding mode was saved changed and restored within each function.

During testing it was discovered that creating the data structures (DataA and DataB) for SIMD multiplication caused a 22% performance decrease, compared to multiplying the bounds in a sequential fashion. As such, two extra algorithms were included: Vec.Raw† and Vec.9-cases‡. Whilst both are fully vectorised, the former operates by producing the initial products via four sequential scalar multiplications and not via SIMD, whilst the latter makes use of Vec.Raw† for its special case; thus both bypass the overheads incurred in the generation of SIMD data structures.

7. Results

Table 5 shows the vital statistics on implementations of the above algorithms. As the performance of the majority of these algorithms are data-dependent and this data was randomly generated, the static code complexity may differ from that of the executed path and to this end, the best and worst case scenarios were recorded. The code complexity was measured in terms of the number of binary operators in the code (comparisons such as =, $\leq$, $>$ etc.), the number of IF/ELSE statements (and if these were nested), the number of integer and floating-point multiplications required and finally the number of variable assignments. An individual call to a SIMD operation is counted as a single operation, regardless of the number of values which it operates upon. These results can be seen to compare favourably to even the double-length products by Wolff von Gudenberg in [8], without requiring the custom hardware suggested.

[3]Gprof is the GNU Graph PROFiler [2].

[4]Family 6, model 7, stepping 3, 500MHz with 1MB L2 cache.

Table 6 reflects the efficiency of the compiled code in terms of the size of the resultant executable, the number of low-level instructions which this contains and the efficiency of the function required to do this. The latter being as the number of μs taken to execute the function, averaged over 1×10^8 calls. Therefore, the less time spent in the function, the greater the efficiency of the algorithm.

8. Conclusion

In conclusion, we have presented a single operation to uniquely identify interval cases (Section 5.2); vectorised formulations and implementations of traditional interval multiplication operations (Sections 5.1 and 5.2); and a means of reducing interval multiplication to only two floating point calculations (Section 4). As can be seen from Table 6, the 9-case method is the fastest, followed (respectively) by the brute-force, vectorised 9-case, vectorised brute-force and integer implementations.

The integer method (described in Section 4) provides the poorest performance as the speed gained by avoiding checking the signs on the bounds and removing the special case does not outweigh the overhead incurred by declaring, using and sorting the additional integer variables. FPU performance has improved considerably in recent years and so we must conclude the integer algorithm would be best suited to systems where there is a high FPU latency.

The performance of the vectorised implementations surpassed the forementioned integer technique and whilst these techniques may look better on paper, it would appear that the costly branching instructions which we strove to remove were in fact replaced with something even more costly, namely: multiple assignments per function. This was highlighted with the vectorised brute-force method (Section 5.1). Setting up data structures dataA and dataB and subsequently performing a SIMD multiply operation on them caused a 22% performance hit, compared to simply declaring something akin to: `product.f[0] =` $\underline{x}$ `*` $\underline{y}$`; product.f[1] =` $\underline{x}$ `*` $\overline{y}$`;` etc. for all four elements in the array *product*. A further decrease in performance was due to the four rotate operations required by the vectorised brute-force method. These are not supported on the Intel architecture upon which it was implemented and so had to be emulated by a sequence of assignments, increasing both program size and latency. This subsequently impacted the vectorised 9-case method which falls back to the brute force algorithm for the 'special case' (case 9).

The original 9-case method proved so successful not only due to its ability to reduce the number of floating-point operations required, but also due to the minimal number of intermediate variables and assignments it requires. As such, the performance decrease incurred by branching was outweighed by the fact the program did not need to pause to

Algorithm	Binary Operations	IF/ELSE statements	Nested IFs	Int. Mult.	FP Mult.	Assign-ments
Brute-Force	6	6	0	0	8	10/17
9-cases	2/6	2/6	1/5	0	2/4	3/7
Integer	28	26	0	4/8	2	22/42
Vec. Raw	4	0	0	0	2	41
Vec. Raw†	4	0	0	0	8	31
Vec. 9-cases	1/5	0	0	0	2	8/46
Vec. 9-cases‡	1/5	0	0	0	2	8/36

Table 5. Code complexity

Algorithm	Program size (Bytes)	Assembled instructions	Time spent in Algorithm (μs)
Brute-Force	15120	75	1.9144
9-cases	15442	127	1.5028
Integer	15730	246	4.5078
Vectorised Raw	15678	99	4.1442
Vectorised Raw†	15249	93	3.4028
Vectorised 9-case	16662	384	2.6940
Vectorised 9-case‡	15249	378	2.3056

Table 6. Code complexity and operations

wait for registers to become available or shuffle much data. The brute-force method was also relatively successful, however despite its simplicity, the additional floating point operations and comparisons reduced its performance compared to the 9-case algorithm.

Whilst we were unable to improve upon the standard 9-case method, we were able to provide a fully vectorised algorithm (vectorised 9-case algorithm, described in Section 5.2) which provided similar performance to the brute-force algorithm. Given the current trends in architecture evolution and some further work, we believe it is an interesting approach and may prove more successful in the future.

These approaches differ in several ways to that of previous work such as Wolff von Gudenberg's SIMD-aware interval arithmetic system [8]. Though the techniques we have outlined above also include counts of the multiply/compare operations required per system; we have included a novel means of interval multiplication and made use of the available SIMD hardware to vectorise the brute-force and 9-case operations. Furthermore, the above algorithms have been implemented in a real-world environment thus providing performance statistics whilst taking into account the overheads incurred. These implementations have made use of SIMD technology without resorting to custom interval-acceleration hardware such as unrounded floating-point modes.

9. Further work

While the performance of the described algorithms has been disappointing when compared to the 9-case method, it should be borne in mind that this technique has had many years to optimised. We are still convinced that vectorisation is a worthy cause and there is still scope for optimisation in these algorithms. Scalar processors have had a variety of rotate operations for many years and the Altivec SIMD architecture enhancements already support this. As SIMD architectures evolve many other processors will undoubtedly provide similar support, allowing many of the assignments used in the vectorised brute-force method to be replaced with a single instruction. This will not only improve performance for the vectorised brute-force method, but also the vectorised 9-case function which relies upon it for its 'special case'. Acceleration via the property $\nabla(value) = -(\triangle(-value))$ will also be investigated to reduce the number of changes to the rounding mode from 3 (up, down, restore) to only two (up and restore). Finally, given the fixed number of operations to compute the product of two intervals via the integer method, we expect this to lend itself to a hardware implementation and are working towards this.

References

[1] D. Chiriaev and G. W. Walster. Interval arithmetic specification. www.mscs.mu.edu/ globsol/Papers/spec.ps, 1998.

[2] J. Fenlason and R. Stallman. *GNU gprof - The GNU Profiler*. The Free Software Foundation, 59 Temple Place - Suite 330, Boston, MA 02111, USA, 1998.

[3] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[4] I. Ollmann. Altivec (a.k.a velocity engine). Technical report, Caltech, 2001.

[5] M. Schulte, K. Bickerstaff, and E. Swartzlander. Hardware interval multipliers. *Journal of Theoretical and Applied Informatics*, 3(2):73–90, 1996.

[6] M. J. Schulte and E. E. Swartzlander. Software and hardware techniques for accurate, self-validating arithmetic. In *Applications of Interval Computations*, pages 381–404, 1996.

[7] J. E. Stine and M. J. Schulte. A combined interval and floating point multiplier. In *Great Lakes Symposium on VLSI*, pages 208–215. IEEE Computer Society, 1998.

[8] J. Wolff von Gudenberg. Hardware support for interval arithmetic. In G. Alefeld, A. Frommer, and B. Lang, editors, *Scientific Computing and Validated Numerics: Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics - SCAN '95*, pages 32–37, Berlin, 1996. Akademie Verlag.

On the interval Gaussian algorithm

Günter Mayer
Universität Rostock
Institut für Mathematik
Universitätsplatz 1, 18055 Rostock, Germany
guenter.mayer@uni-rostock.de

Abstract

We give a survey on criteria for the feasibility and non–feasibility of the interval Gaussian algorithm. In particular, we consider generalized diagonally dominant matrices, appropriate sparse matrices, and Hessenberg matrices. Moreover, we recall alternative representations and pivoting.

1. Introduction

In order to verify and to enclose solutions of linear systems $Ax = b$ a variety of methods are available as can be seen from any text book or survey on interval computations such as [4], [7], [31], [40], or [44]. This remark refers also to the solution set Σ of linear systems if $A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^n$ are allowed to vary within interval quantities $[A]$ and $[b]$, respectively. One such method is the interval Gaussian algorithm which constructs an interval vector $[x]^G = \mathrm{IGA}([A],[b])$ such that

$$\Sigma = \{\, x \mid Ax = b,\ A \in [A],\ b \in [b] \,\} \subseteq [x]^G$$

holds. Roughly speaking the algorithm uses the formulae of the conventional Gaussian algorithm and replaces the real entries and operations by intervals and corresponding interval operations (cf. Section 2). Since, in general, the set Σ is not an interval (cf. [44]), and also due to data dependency, $[x]^G$ often overestimates this set. But there are also classes of input data $[A],\ [b]$ such that $[x]^G$ results in the interval hull of Σ – at least if roundings are excluded; cf. for instance [11]. The conventional Gaussian algorithm may break down by division by zero. If this failure does not occur we call the algorithm feasible, otherwise infeasible. This terminology is also used for the interval Gaussian algorithm where 'division by zero' is replaced by 'division by an interval which contains zero'.

The feasibility of the conventional Gaussian algorithm is guaranteed by the following necessary and sufficient condition which can be found, for instance, in [47]:

Theorem 1 *Let $A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^n$. Then the Gaussian algorithm is feasible if and only if all leading principal submatrices of A are non–singular.*

Unfortunately, a similar criterion is missing for the interval version of the algorithm. Assuming that the interval Gaussian algorithm is feasible if and only if it is for every pair $(A,b) \in [A] \times [b]$ was shown to be false by a famous 3×3 counterexample due to Reichmann [49]. Up to now only necessary *or* sufficient global criteria are known for the existence of $[x]^G$. It is the purpose of this paper to summarize them in order to create a platform from which one can start for new ones. Emphasize was laid in creating categories which should help to order the criteria. Here the reviews [14] and [35] were helpful – at least up to the date of their appearance. We add, of course, all newer criteria known to us hoping not to have missed too many.

The algorithm was suggested already by Dwyer [13]. Moore [40] presented the formulae for 2×2 interval matrices while Hansen and Smith [29], Apostolatos and Kulisch [10], Nickel [45], Wißkirchen [60] mentioned the algorithm or partly worked with the interval bounds. Alefeld and Herzberger [3] listed the formulae of Section 2 explicitly. Starting with their appearance in 1974 and lasting up to now a variety of contributions were made on the subject. They contain a lot of techniques applicable also in other fields of interval analysis. In so far, the results sometimes might have a purely theoretical touch. The ideas, however, are interesting and useful.

Our paper is organized as follows: First we repeat the essential formulae of the algorithm – mainly for notational reasons. We also recall two alternatives which are due to Schwandt [57] and Neumaier [42]. In Section 3 we list criteria of feasibility which are based on generalized diagonally dominant matrices, among them H– and M–matrices. Section 4 is devoted to perturbations and Section 5 considers particular sparse matrices. Section 6 deals with Hessenberg matrices. It contains a new formulation of Reichmann's result in [48] and a representation of the possible structure of the matrices. Section 7 collects some remarks

on pivoting while Section 8 presents criteria of infeasibility. Section 9 contains some final remarks. By lack of space we list all the theorems without proof but with references.

2. Representations

We first describe our notation. We use square brackets for intervals $[a] = [\underline{a}, \overline{a}]$, and $[a]_{ij},\ [b]_i$ for the entries of interval matrices $[A]$ and interval vectors $[b]$, respectively. We denote by $\mathbb{IR},\ \mathbb{IR}^n$, and $\mathbb{IR}^{n\times n}$ the set of intervals, interval vectors with n components and $n \times n$ interval matrices. By $e^{(k)}$ we mean the k–th column of the $n \times n$ identity matrix I. We equip $\mathbb{R}^n$ and $\mathbb{R}^{n\times n}$ with the usual entrywise defined partial ordering $\leq$ and write $x < y$ for vectors if strict inequality holds for all components. For intervals $[a]$ we define the mignitude $\langle [a] \rangle = \min\{\, |a| \mid a \in [a]\,\}$, the absolute value $|[a]| = \max\{\, |\underline{a}|, |\overline{a}|\,\}$, the midpoint $\mathrm{mid}([a]) = (\underline{a} + \overline{a})/2$ and the diameter $d([a]) = \overline{a} - \underline{a}$. With $[A] \in \mathbb{IR}^{n\times n}$ we associate the Ostrowski matrix $\langle [A] \rangle = (c_{ij}) \in \mathbb{R}^{n\times n}$ which is given by $c_{ii} = \langle [a]_{ii} \rangle$ and $c_{ij} = -|[a]_{ij}|$ if $i \neq j$. In addition, we will use the absolute value $|[A]| = (|[a]_{ij}|) \in \mathbb{R}^{n\times n}$ of an interval matrix $[A]$. We call $[A] \in \mathbb{IR}^{n\times n}$ irreducible if $|[A]|$ is irreducible. The spectral radius of a matrix $A \in \mathbb{R}^{n\times n}$ is denoted by $\rho(A)$.

The interval Gaussian algorithm starts with $[A]^{(1)} = [A] \in \mathbb{IR}^{n\times n},\ [b]^{(1)} \in \mathbb{IR}^n$ and – if it is feasible – results in a final vector $[x]^G = \mathrm{IGA}([A], [b]) \in \mathbb{IR}^n$ via intermediate quantities $[A]^{(k)},\ [b]^{(k)},\ k = 2, \ldots, n$, in the following way:

$$[a]_{ij}^{(k+1)} = \begin{cases} [a]_{ij}^{(k)}, & i = 1, \ldots, k, \\ & j = 1, \ldots, n, \\ [a]_{ij}^{(k)} - \dfrac{[a]_{ik}^{(k)} [a]_{kj}^{(k)}}{[a]_{kk}^{(k)}}, & i, j = k+1, \ldots, n, \\ 0 & \text{otherwise} \end{cases}$$

$$[b]_i^{(k+1)} = \begin{cases} [b]_i^{(k)}, & i = 1, \ldots, k, \\ [b]_i^{(k)} - \dfrac{[a]_{ik}^{(k)}}{[a]_{kk}^{(k)}} [b]_k^{(k)}, & i = k+1, \ldots, n, \\ & k = 1, \ldots, n-1, \end{cases}$$

$$[x]_i^G = \Big([b]_i^{(n)} - \textstyle\sum_{j=i+1}^n [a]_{ij}^{(n)} [x]_j^G \Big) \,/\, [a]_{ii}^{(n)}, \quad i = n, n-1, \ldots, 1.$$

In [57] Schwandt presented the multiplicative representation

$$\begin{aligned} [x]^G \;=\; & [D]^{(1)} ([U]^{(1)} ([D]^{(2)} ([U]^{(2)} (\ldots ([U]^{(n-1)} \times \\ & ([D]^{(n)} ([L]^{(n-1)} (\ldots ([L]^{(2)} ([L]^{(1)} [b])) \ldots), \end{aligned} \tag{1}$$

where

$$\begin{aligned} [D]^{(k)} &= I - e^{(k)} (e^{(k)})^T (1 - \frac{1}{[a]_{kk}^{(k)}}), \\ [L]^{(k)} &= I - \Big(e^{(k)} (0, \ldots, 0, [a]_{k+1,k}^{(k)}, \ldots, [a]_{nk}^{(k)}) / [a]_{kk}^{(k)} \Big)^T, \\ [U]^{(k)} &= I - e^{(k)} (0, \ldots, 0, [a]_{k,k+1}^{(k)}, \ldots, [a]_{kn}^{(k)}) \,; \end{aligned}$$

cf. also [2]. Neumaier obtained $[x]^G$ in [42], [44] recursively via the partition

$$[A] = \begin{pmatrix} [a]_{11} & [c]^T \\ [d] & [A'] \end{pmatrix}, \quad [c],\, [d] \in \mathbb{IR}^{n-1},$$

and the Schur complement $S_{[A]} = [A'] - [d][c]^T / [a]_{11} \in \mathbb{IR}^{(n-1)\times(n-1)}$, provided that $0 \notin [a]_{11}$. He defines the triangular decomposition $([L], [U])$ of $[A]$ to exist, if either $n = 1,\ [L] = 1,\ [U] = [A] \not\ni 0$ or

$$[L] = \begin{pmatrix} 1 & 0 \\ [d]/[a]_{11} & [L'] \end{pmatrix}, \quad [U] = \begin{pmatrix} [a]_{11} & [c]^T \\ 0 & [U'] \end{pmatrix},$$

where $0 \notin [a]_{11}$ and where $([L'], [U'])$ is the triangular decomposition of $S_{[A]}$. With the decomposition $[b] = ([\beta], [b']^T)^T,\ [\beta] \in \mathbb{IR},\ [b'] \in \mathbb{IR}^{n-1}$ he finally ends up with the recursion

$$[x]^G = \mathrm{IGA}([A], [b]) = \begin{pmatrix} [x]_1^G \\ [x']^G \end{pmatrix},$$

where

$$\begin{aligned} [x']^G &= \mathrm{IGA}(S_{[A]}, [b'] - [\beta][d]/[a]_{11}), \\ [x]_1^G &= ([\beta] - [c]^T [x']^G) / [a]_{11} \,. \end{aligned}$$

If the triangular decomposition $([L], [U])$ exists $[x]^G$ can equivalently be expressed by

$$[x]^G = \mathrm{IGA}\big([U], \mathrm{IGA}([L], [b])\big).$$

3. Generalized diagonally dominant matrices

Our first class of matrices leads us to a definition which was introduced in [41] for real matrices.

Definition 1 *A matrix $[A] \in \mathbb{IR}^{n\times n}$ is called generalized diagonally dominant, if there is a real vector $u > 0$ such that $\langle [A] \rangle u \geq 0$. If, in addition, $[A]$ is irreducible and $(\langle [A] \rangle u)_i > 0$ holds for at least one component i then $[A]$ is called generalized irreducibly diagonally dominant. If u can be chosen to be $u = e = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$ then the specification 'generalized' can be dropped.*

Note the slight difference between 'irreducibly diagonally dominant' and 'irreducible and diagonally dominant' which

is relevant for our first necessary and sufficient criterion given in [36]. In order to formulate it we need the sign matrix $S \in \mathbb{R}^{n\times n}$ of $[A]$ which is defined by $s_{ij} = \text{sign}(\,\text{mid}([a]_{ij})\,)$, and the extended sign matrix $S' \in \mathbb{R}^{n\times n}$ which is constructed by means of S and the following algorithm:

$S' := S$
for $k := 1$ to $n-1$ do
  for $i := k+1$ to n do
    for $j := k+1$ to n do
      if $s'_{ij} = 0$ then $s'_{ij} := -s'_{ik}s'_{kk}s'_{kj}$.

Theorem 2 *Let $[A] \in \mathbb{IR}^{n\times n}$ be irreducible and generalized diagonally dominant. Moreover, let $[b] \in \mathbb{IR}^n$ and S' be the extended sign matrix . Then $[x]^G$ exists if and only if $[A]$ is generalized irreducibly diagonally dominant or if*

$$s'_{ij}\, s'_{ik}\, s'_{kk}\, s'_{kj} = \begin{cases} 1, & \text{if } i \neq j \\ -1, & \text{if } i = j \end{cases} \tag{2}$$

holds for some triple (i,j,k) with $k < i, j$.

In order to apply Theorem 2 to matrices which are not necessarily irreducible we introduce the so–called reducible normal form of $[A]$. To this end let P be a permutation matrix such that

$$R(|[A]|) = P|[A]|P^T = \begin{pmatrix} R_{11} & O & \dots & O \\ R_{21} & R_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & O \\ R_{r1} & R_{r2} & \dots & R_{rr} \end{pmatrix}$$

is 'the' reducible normal form of $|[A]|$ defined for real matrices as in [19], e.g. Here, $R_{ss},\ s = 1,\dots,r$, are square diagonal blocks which are 1×1 zero matrices or irreducible. It is known that the reducible normal form is unique up to permutations of rows and columns within the block rows and block columns and up to permutations of specific block rows and block columns. With the same permutation matrix P as above the matrix $R([A]) = P[A]P^T$ is called reducible normal form of $[A]$ with blocks $[R]_{st}$.

Let π be the permutation associated with P and let $i_1,\ i_2$ be any two indices of $[A]$ whose images $\pi(i_1),\ \pi(i_2)$ belong to the same block $[R]_{ss}$ of $R([A])$. If $i_1 < i_2$ always implies $\pi(i_1) < \pi(i_2)$ then we call P order preserving within blocks.

With this terminology we can formulate our next theorem.

Theorem 3 *Let $[A] \in \mathbb{IR}^{n\times n}$, $[b] \in \mathbb{IR}^n$ and let $P \in \mathbb{R}^{n\times n}$ be a permutation matrix such that $R([A]) = ([R]_{st}) = P[A]P^T$ is 'the' reducible normal form of $[A]$ with blocks $[R]_{st},\ s,t = 1,\dots,r$. Assume that P is order preserving within blocks. Then the following properties are equivalent.*

(i) IGA$([A],[b])$ *exists.*

(ii) IGA$(P[A]P^T, P[b]) =$ IGA$(R([A]), P[b])$ *exists.*

(iii) IGA$([R]_{ss}, (P[b])_s)$ *exists for $s = 1,\dots,r$, where $(P[b])_s$ denotes the part of $P[b]$ which corresponds to $[R]_{ss}$.*

Theorem 3 may become false if P is not order preserving within blocks as is shown in [36] by an example.

Based on this result Theorem 2 can be generalized to matrices whose Ostrowski matrix may be reducible. Partly by virtue of this generalization we get the following sufficient condition for which we recall that a real matrix $A \in \mathbb{R}^{n\times n}$ is an M–matrix if $a_{ij} \leq 0$ for $i \neq j$, $\det(A) \neq 0$ and $A^{-1} \geq O$. An interval matrix $[A] \in \mathbb{IR}^{n\times n}$ is called an M–matrix if all matrices $A \in [A]$ are M–matrices. It is an H–matrix if $\langle[A]\rangle$ is an M–matrix. Since M–matrices are particular H–matrices and since the latter ones are particular generalized diagonally dominant matrices (cf. [12]), parts of Theorem 4 are not very astonishing.

Theorem 4 *Let $[A] \in \mathbb{IR}^{n\times n},\ n \geq 2$, and let $[b] \in \mathbb{IR}^n$. In each of the following cases $[x]^G$ exists.*

a) $[A]$ *is an M–matrix; cf. [3], [11].*

b) $[A]$ *is an H–matrix; cf. [1].*

c) $[A]$ *is generalized σ diagonally dominant, where σ can be replaced by any of the following specifications: 'strictly', 'irreducibly', 'Ω–'. (For a definition in the first and the latter case see [36].)*

d) $[A]$ *is generalized diagonally dominant and $\langle[A]\rangle$ is non–singular.*

e) $[A]$ *is irreducible and generalized diagonally dominant, and (2) holds with S instead of S'.*

Theorem 4d) follows since $\langle[A]\rangle + \varepsilon I$ are M–matrices for all positive values ε. Hence $(\langle[A]\rangle + \varepsilon I)^{-1} \geq O$, and, as a limit, $\langle[A]\rangle^{-1} \geq O$ holds, too. Thus $[A]$ is an H–matrix and Theorem 4b) applies.

If one drops the property 'generalized' in Theorem 2 or 4 one ends up with results in [15], [16] and [37].

In [36] and [38] the following necessary and sufficient condition was proved:

Theorem 5 *Let $[A] = I + [-R,R] \in \mathbb{IR}^{n\times n}$, $O \leq R \in \mathbb{R}^{n\times n}$, $[b] \in \mathbb{IR}^n$. Then $[x]^G$ exists if and only if $\rho(R) < 1$ which in turn is equivalent to $[A]$ being an H–matrix.*

Although in Theorem 5 the structure of $[A]$ seems to be very specific one can always obtain it from a general interval matrix $[A]$ without singular element matrices. Just precondition $[A]$ and $[b]$ by the midpoint inverse $C = (\mathrm{mid}([A]))^{-1}$. But note that by data dependencies $C[A]$ may contain singular matrices, whence the interval Gaussian algorithm breaks down for $C[A]$ while it could have been feasible for the original matrix $[A]$. As an example in this direction consider

$$[A] = \begin{pmatrix} [1,3] & [-3,-1] \\ [1,3] & [1,3] \end{pmatrix}$$

with

$$C[A] = \frac{1}{2} \begin{pmatrix} [1,3] & [-1,1] \\ [-1,1] & [1,3] \end{pmatrix} .$$

4. Perturbations

If $[x]^G = \mathrm{IGA}([A],[b])$ exists for some matrix $[A]$ it is clear by the continuity of the interval arithmetic that $\mathrm{IGA}([B],[c])$ exists for arbitrary vectors $[c]$ and matrices $[B]$ which are sufficiently close to $[A]$, where 'close' refers to the usual Hausdorff metric $q([a],[b]) = \max\{\, |\underline{a} - \underline{b}|,\, |\overline{a} - \overline{b}| \,\}$ which is used to define the matrix $q([A],[B]) = (q([a]_{ij},[b]_{ij})) \in \mathbb{R}^{n\times n}$. Neumaier quantified this observation in [43], [44]:

Theorem 6 *Let* $[A],\ [B] \in \mathbb{IR}^{n\times n},\ [b] \in \mathbb{IR}^n$. *If* $[x]^G = \mathrm{IGA}([A],[b])$ *exists and if* $\rho(|[A]^G|q([A],[B])) < 1$ *then* $\mathrm{IGA}([B],[c])$ *exists for arbitrary* $[c] \in \mathbb{IR}^n$.

The interval matrix $[A]^G$ is obtained from (1) if one replaces there $[b]$ by the matrix $I \cdot [-1,1]$.

5. Particular sparse matrices

In this section we consider tridiagonal matrices and related ones. In order to characterize these matrices we use the concept of an undirected graph of a real matrix $A \in \mathbb{R}^{n\times n}$ with the nodes $1,\ldots,n$; cf. for instance [25]. We call j a neighbor of the node i $(\neq j)$ if i and j are connected by an edge. The number of neighbors of i are the degree of i in the underlying graph. Let G_k denote the k–th elimination graph of $[A]$, i.e., the undirected graph of $|[A]^{(k)}|$ in which the nodes $1,\ldots,k-1$ and the corresponding edges have been removed and for which we assume that $[a]_{ij}^{(k-1)} \neq 0$ implies $[a]_{ij}^{(k)} \neq 0,\ i,j \geq k$ (no accidental zeros!); cf. [25]. If in G_k the node k has the smallest degree and if this holds for all $k = 1,\ldots,n$ then we say that $[A]$ is ordered by minimum degree. If the graph of such a matrix has tree structure (i.e., there are no cycles of length ≥ 3) the following result holds.

Theorem 7 *Let* $[A] \in \mathbb{IR}^{n\times n},\ [b] \in \mathbb{IR}^n$. *If the (undirected) graph of* $|[A]|$ *is a tree which is ordered by minimum degree then* $[x]^G$ *exists if and only if* x^G *exists for all matrices* $A \in [A]$.

Theorem 7 was published in [14] and contains Reichmann's condition in [49] for tridiagonal matrices, Garloff's theorem in [22] on tridiagonal matrices which contain only regular and totally non–negative element matrices, and Schäfer's result in [53] and [54] on arrowhead matrices.

6. Hessenberg matrices

In [48] Reichmann derives a sufficient criterion for interval upper Hessenberg matrices for which we use the following notation

$$\sigma([a]) \quad = \quad \begin{cases} 1 & \text{if } \underline{a} \geq 0 \\ -1 & \text{if } \overline{a} \leq 0 \\ 0 & \text{otherwise} \end{cases} , \quad [a] \in \mathbb{IR}\backslash\{0\} .$$

Theorem 8 *Let* $[b] \in \mathbb{IR}^n,\ n \geq 2$, *and let* $[A] \in \mathbb{IR}^{n\times n}$ *be an upper Hessenberg matrix satisfying* $0 \notin [a]_{ii},\ i = 1,\ldots,n$ *and* $[a]_{i+1,i} \neq 0,\ i = 1,\ldots,n-1$. *Then* $[x]^G$ *exists if for each* $i = 1,\ldots,n-1$ *and each* $j = i+1,\ldots,n$ *one of the following two (mutually excluding) conditions holds.*

(i) $[a]_{ij} = 0 \Rightarrow [a]_{pj} = 0,\ p = 1,\ldots,i-1$;

(ii) $[a]_{ij} \neq 0$
$\Rightarrow \sigma([a]_{ii})\sigma([a]_{i+1,j}) = -\sigma([a]_{i+1,i})\sigma([a]_{ij})$.

Theorem 8 looks slightly different from Reichmann's criterion (Satz 4.1) in [48]. If one corrects a typing error in [48] one can easily see the equivalence. (For the typing error compare condition (2) I) of Satz 4.1 in [48] with the first implication in '2. Fall 1:' of its proof. This proof is correct if $[a_{k,i+1}]$ in condition (2) I) is replaced by $[a_{k,j}]$. For the equivalence take into account that the assumption $0 \notin [a]_{ii}$ implies $\underline{a}_{ii} > 0$ or $\overline{a}_{ii} < 0$ depending on $i = 1,\ldots,n$. Therefore, $\sigma([a]_{ii}) = \mathrm{sign}([a]_{ii})$ with 'sign' as defined in [48]. Moreover, from $[a]_{ij} \neq 0$ together with (i) of Theorem 8 we get $[a]_{i+1,j} \neq 0$.)

For upper Hessenberg matrices only one element must be eliminated in each elimination step k. Hence $[A]^{(k)}$ and $[A]^{(k+1)}$ only differ by the row $k+1$. Condition (i) says that with a zero entry all other entries further up in the same column are zero. Condition (ii) allows only particular sign patterns for the matrices $A \in [A]$. It seems to be closely related to (2) if S' is replaced by the sign matrix S in Section 3. But note that S is defined via midpoints while $\sigma([a])$ uses endpoints unless $0 \in [a]$. Moreover, (2) needs to hold only once in contrast to the property in (ii).

The conditions and assumptions of Theorem 8 imply

$$[a]_{ij}^{(k+1)} = 0, \quad \text{if } [a]_{ij}^{(k)} = 0 \text{ and } j \geq i+1$$

and

$$\sigma([a]_{ij}^{(k+1)}) = \sigma([a]_{ij}^{(k)}), \quad \text{if } [a]_{ij}^{(k)} \neq 0 \text{ and } j \geq i+1.$$

Moreover, they yield

$$\sigma([a]_{ii}^{(k+1)}) = \sigma([a]_{ii}^{(k)}) \text{ and } 0 \notin [a]_{ii}^{(k+1)}.$$

This means that the sign pattern (in the sense above) remains fixed in the upper triangle during the whole elimination process. To show this by induction is the crucial part of Reichmann's proof for Theorem 8.

If $[a]_{ij} \neq 0$ for some i, j with $i < j$ then (i) implies

$$[a]_{kj} \neq 0 \text{ for all } k \text{ with } i < k < j. \tag{3}$$

If, in addition, $\sigma([a]_{ij}) = 0$ then $\sigma([a]_{pj}) = 0$, $p = i+1, \ldots, j-1$, by virtue of (ii) and induction. In particular, $\sigma([a]_{j-1,j}) = 0$. This contradicts (ii) for $i = j-1$ since $\sigma([a]_{j-1,j-1})\sigma([a]_{jj}) \neq 0$. Hence if the assumptions of Theorem 8 are fulfilled no entry in the upper triangle of $[A]$ contains zero in its interior, i.e., either $[a]_{ij} = 0$ or $\sigma([a]_{ij}) \neq 0$ holds for $i \leq j$. Using (ii) if $j = i+1$, and (ii) and (3) with $k = i+1$ if $j > i+1$ we also get $\sigma([a]_{i+1,i}) \neq 0$ provided that $[a]_{ij} \neq 0$. Thus $\sigma([a]_{i+1,i}) = 0$ can only occur if $[a]_{ij} = 0$, $j = i+1, \ldots, n$, whence by (i) the whole block $([a]_{kj})_{k=1,\ldots,i,\ j=i+1,\ldots,n}$ is zero.

Assume now $\sigma([a]_{ii}) = 1$, $i = 1, \ldots, n$ (in addition to $0 \notin [a]_{ii}$) which can always be obtained by an appropriate scaling. Then the non–zero sign pattern of $[A]$ is completely controlled by the signs of the entries in the first lower subdiagonal. This can be seen for instance when starting with the last row and filling up the sign pattern row by row. In order to illustrate the possible sign structure we assume for simplicity that no entry in the upper triangle is zero. Then $\sigma([a]_{i+1,i}) \neq 0$, $i = 1, \ldots, n-1$. Precondition now $[A]$ by a signature matrix $D = \operatorname{diag}(d_1, \ldots, d_n)$, $d_i = \pm 1$, from the left such that $\sigma([a]_{ii}) = 1$, $i = 1, \ldots, n$, as above. Then precondition from the left and the right by a signature matrix $\hat{D} = \operatorname{diag}(\hat{d}_1, \ldots, \hat{d}_n)$ in the following recursive way: $\hat{d}_1 = \pm 1$ is arbitrary. If $\hat{d}_1, \ldots, \hat{d}_i$ are known choose $\hat{d}_{i+1} = \pm 1$ such that $\sigma(\hat{d}_{i+1} d_{i+1} [a]_{i+1,i} \hat{d}_i) = -1$. Consider now $[\hat{A}] = \hat{D}D[A]\hat{D}$. Since $\sigma([\hat{a}]_{ii}) = 1 = -\sigma([\hat{a}]_{i+1,i})$ condition (ii) reads $\sigma([a]_{i+1,j}) = \sigma([a]_{ij})$, $j = i+1, \ldots, n$, from which we necessarily get the sign pattern

$$\begin{pmatrix} + & + & + & \ldots & + \\ - & + & + & \ldots & + \\ 0 & - & + & \ldots & + \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & - & + \end{pmatrix}, \tag{4}$$

i.e., the signs of the entries of each $\tilde{A} \in [\hat{A}]$ are nonpositive in the first lower subdiagonal, positive in the diagonal, and nonnegative in the whole strict upper triangle. Therefore, if $[A]$ fulfills the assumptions of Theorem 8 and has no zero in the upper triangle then the sign pattern of its element matrices grows out from (4) changing there the signs according to a multiplication of $[A]$ with a signature matrix D_1 from the left and a signature matrix D_2 from the right. Note that $\operatorname{IGA}([A], [b])$ exists if and only if $\operatorname{IGA}(D_1[A]D_2, D_1[b])$ exists, where D_1, D_2 are arbitrary non–singular diagonal matrices from $\mathbb{R}^{n \times n}$; cf. [36] or [38].

It is obvious that Theorem 8 can also be applied to irreducible tridiagonal matrices (cf. [48]): Assume again $0 \notin [a]_{ii}$, $i = 1, \ldots, n$, and $[a]_{i+1,i} \neq 0$, $i = 1, \ldots, n-1$. Then by Theorem 8 $[x]^G$ exists if $\sigma([a]_{ii})\sigma([a]_{i+1,i+1}) = -\sigma([a]_{i+1,i})\sigma([a]_{i,i+1})$ whenever $[a]_{i,i+1} \neq 0$.

For tridiagonal matrices condition (i) of Theorem 8 is always fulfilled.

In [50] M–matrices, H–matrices and upper Hessenberg matrices are combined in a simple way to obtain larger ones for which the existence of $[x]^G$ is shown.

It is tempting to generalize condition (ii) to arbitrary (i.e., non–Hessenberg) matrices via

$$\begin{aligned} &[a]_{ij} \neq 0 \\ &\Rightarrow \sigma([a]_{ii})\sigma([a]_{pj}) = -\sigma([a]_{pi})\sigma([a]_{ij}),\ p, j \geq i+1. \end{aligned}$$

Already 3×3 examples with $0 \notin [a]_{ij}$, $i, j = 1, 2, 3$, show, however, that this attempt must fail in general.

7. Pivoting

Up to now pivoting is not very well studied for the interval Gaussian algorithm. Wongwises suggests in [61] to choose as pivot in $[A]^{(k)}$ the entry $[a]_{pk}^{(k)}$ which satisfies

$$\langle [a]_{pk}^{(k)} \rangle = \max \{ \langle [a]_{ik}^{(k)} \rangle \mid k \leq i \leq n \}. \tag{5}$$

If several entries share this property she decides for that one among them which has the smallest diameter. In [30] Hebgen selects the pivot according to the ratio

$$\begin{aligned} &d([a]_{pk}^{(k)}) \,/\, \langle [a]_{pk}^{(k)} \rangle \\ &\quad = \min \{ d([a]_{ik}^{(k)}) \,/\, \langle [a]_{ik}^{(k)} \rangle \mid k \leq i \leq n,\ 0 \notin [a]_{ik}^{(k)} \}. \end{aligned} \tag{6}$$

In order not to stick automatically at the original entry $[a]_{kk}^{(k)}$ if it is degenerate Hebgen proposes for this case the entry which is degenerate and comes closest to one.

With Ratschek's χ–function

$$\chi([a]) = \begin{cases} \underline{a}/\overline{a} & \text{if } |\underline{a}| \leq |\overline{a}|, \\ \overline{a}/\underline{a} & \text{if } |\underline{a}| > |\overline{a}|, \\ 0 & \text{if } \underline{a} = \overline{a} = 0 \end{cases}$$

one gets $d([a])/\langle[a]\rangle = \chi([a])^{-1} - 1$ as long as $0 \notin [a]$. Therefore, condition (6) can equivalently be expressed by

$$\chi([a]_{pk}^{(k)}) = \max\{\, \chi([a]_{ik}^{(k)}) \;\big|\; k \le i \le n,\ 0 \notin [a]_{ik}^{(k)}\}\,.$$

Note that (6) is invariant with respect to scaling, i.e., with respect to multiplying rows by real numbers $\neq 0$. In [61] it is reported that in many cases (5) yields smaller widths for $[x]^G$ than (6).

8. Infeasibility

For many years only criteria for the *feasibility* of the interval Gaussian algorithm were interesting. In [36] Reichmann's example $[A] \in \mathbb{IR}^{3\times 3},\ [a]_{ii} = 1$ for $i = 1,\ 2,\ 3,$ $[a]_{ij} = [0, \gamma]$ for $i \neq j$ and $\gamma = (-1+\sqrt{5})/2$ (cf. [48]) was the starting point to look for non–trivial criteria which guarantee the failure of the algorithm. By 'non–trivial' we mean interval matrices $[A]$ for which the conventional Gaussian algorithm is feasible for each element matrix $A \in [A]$. Our final theorem is a first result in this direction. It contains Reichmann's example and Neumaier's one in [44] as particular cases.

Theorem 9 *Let $[A] \in \mathbb{IR}^{n\times n}$ contain I with $\underline{a}_{ii} = 1,\ i = 1, \ldots, n$. Choose $c_{ik} \in \{\underline{a}_{ik}, \overline{a}_{ik}\},\ c_{ki} \in \{\underline{a}_{ki}, \overline{a}_{ki}\}$ such that*

$$c_{ik}c_{ki} = \max\{\underline{a}_{ik}\underline{a}_{ki}, \overline{a}_{ik}\overline{a}_{ki}\} \quad \textit{für } i > k,\ k = 1, \ldots, n-1.$$

Define

$$c_{kk}^{(k)} = 1 - \sum_{j=1}^{k-1} \frac{c_{kj}c_{jk}}{c_{jj}^{(j)}}\,, \quad k = 1, 2, \ldots, n,$$

as long as $c_{k-1,k-1}^{(k-1)} > 0$.

If $c_{kk}^{(k)} \le 0$ for some k, then $[x]^G$ does not exist.

Note that the assumption $\underline{a}_{ii} = 1$ is not very restrictive since it can always be fulfilled by preconditioning $[A]$ with an appropriate diagonal point matrix, provided that $0 \notin [a]_{ii},\ i = 1, \ldots, n$ and $0 \in [a]_{ij},\ i \neq j$. Even if these two latter conditions do not hold $I \in [A]$ can be fulfilled by preconditioning $[A]$ with its midpoint inverse $C = (\mathrm{mid}([A]))^{-1}$ with the same remark as for Theorem 5.

Unfortunately, the converse of Theorem 9 does not hold as one can see by the example $[a]_{ii} = 1,\ [a]_{ij} = [-\delta, \delta]$ for $i \neq j,\ 1/2 \le \delta < \gamma,\ \gamma$ as above.

9. Final remarks

Up to now we did not discuss the interval Gaussian algorithm under the aspect of quality of enclosure. Here, the papers [11], [27], [33], [45], [46], [51], [52], [56] might be of interest. We either did not consider the computation of $[x]^G$ when taking into account rounding errors. Some remarks on this topic can be found in [48], [56], [62]. Applications, extensions and modifications of the interval Gaussian algorithm can be found for instance in [2] (Newton's method), [9] (Krawczyk–like method), [17] (parallel interval multisplitting method), [18] (Lanczos process), [24] (block Gauss elimination), [32] (interval input–output analysis in economics), [34] (interval iterative methods), [53] (linear complementarity problem), [57], [58] (Buneman algorithm), [59] (cyclic reduction). This list is by no means complete. In [28] and [39] ideas are presented with which one tries to overcome the early break down of the interval Gaussian algorithm by division by zero. In [8] the algorithm was presented at an elementary level. Preconditioning in connection with the interval Gaussian algorithm is considered for instance in [26], [27], [44], [51].

We finally remark that similar results as in the present paper for the interval Gaussian algorithm can also be derived for the interval Cholesky method introduced in [5] in order to enclose the symmetric solution set

$$\Sigma_{\mathrm{sym}} = \{x \in \mathbb{R}^n \mid Ax = b,\ A = A^T \in [A],\ b \in [b]\}$$

for $[A] = [A]^T$. Here we assume that all *symmetric* element matrices $A \in [A]$ are positive definite. For details see [5], [6], [14], and [55].

Further structured matrices $A \in [A]$ are studied in [20], [21] (circulant matrices) and in [23] (Toeplitz matrices using Trench's and Bareiss' algorithm).

References

[1] G. Alefeld. Über die Durchführbarkeit des Gaußschen Algorithmus bei Gleichungen mit Intervallen als Koeffizienten. *Computing Suppl.*, 1:15–19, 1977.

[2] G. Alefeld. On the convergence of some interval–arithmetic modifications of Newton's method, *SIAM J. Numer. Anal.*, 21:363–372, 1984.

[3] G. Alefeld, J. Herzberger. *Einführung in die Intervallrechnung*. Reihe Informatik 12, Bibliographisches Institut, Mannheim, 1974.

[4] G. Alefeld, J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[5] G. Alefeld, G. Mayer. The Cholesky method for interval data. *Linear Algebra Appl.*, 194:161 – 182, 1993.

[6] G. Alefeld, G. Mayer. On the symmetric and unsymmetric solution set of interval systems. *SIAM J. Matrix Anal. Appl.*, 16:1223–1240, 1995.

[7] G. Alefeld, G. Mayer. Interval Analysis: Theory and Applications. *J. Comp. Appl. Math.*, 121:421–464, 2000. Special Issue: *Numerical Analysis in the 20th Century, Vol. I. Approximation Theory*. Eds.: L. Wuytack, J. Wimp.

[8] G. Alefeld, G. Mayer. The Gaussian algorithm for linear systems with interval data. In *Linear Algebra Gems: Assets for Undergraduate Mathematics*. Eds.: D. Carlson, C. R. Johnson, D. C. Lay, A. D. Porter. The Mathematical Association of America, MAA Notes # 59, 2001, pp. 197–204.

[9] G. Alefeld, L. Platzöder. A quadratically convergent Krawczyk–like algorithm. *SIAM J. Numer. Anal.*, 20:210–219, 1983.

[10] N. Apostolatos, U. Kulisch. Grundzüge einer Intervallrechnung für Matrizen und einige Anwendungen. *Elektron. Rechenanl.*, 10:73–83, 1968.

[11] W. Barth, E. Nuding. Optimale Lösung von Intervallgleichungssystemen. *Computing*, 12:117–125, 1974.

[12] A. Berman, R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Classics in Applied Mathematics 9, SIAM, Philadelphia, 1994.

[13] P. S. Dwyer. *Linear Computations*. Wiley, New York, 1951.

[14] A. Frommer. A feasibility result for interval Gaussian elimination relying on graph structure. In *Symbolic Algebraic Methods and Verification Methods*. Eds.: G. Alefeld, J. Rohn, S. Rump, T. Yamamoto. SpringerMathematics, Springer, Wien, 2001, pp. 79–86.

[15] A. Frommer, G. Mayer. A new criterion to guarantee the feasibility of the interval Gaussian algorithm. *SIAM J. Matrix Anal. Appl.*, 14:408–419, 1993.

[16] A. Frommer, G. Mayer. Linear systems with Ω–diagonally dominant matrices and related ones. *Linear Algebra Appl.*, 186:165–181, 1993.

[17] A. Frommer, G. Mayer. Parallel interval multisplittings. *Numer. Math.*, 56:255–267, 1989.

[18] A. Frommer, A. Weinberg. Verified error bounds for linear systems through the Lanczos Process. *Reliable Computing*, 5(3):255–267, 1999.

[19] F. S. Gantmacher. *Matrizentheorie*. Springer, Berlin, 1986.

[20] J. Garloff. Untersuchungen zur Intervallinterpolation. Thesis, Universität Freiburg, Institut für Angewandte Mathematik, *Freiburger Intervall–Berichte*, 80/5, 1980.

[21] J. Garloff. Zur intervallmäßigen Durchführung der schnellen Fourier–Transformation. *Z. Angew. Math. Mech.*, 60:T291–T292, 1980.

[22] J. Garloff. Totally nonnegative interval matrices. In *Interval Mathematics 1980*. Ed.: K. L. E. Nickel. Academic Press, New York, 1980, pp. 317–327.

[23] J. Garloff. Solution of linear equations having a Toeplitz interval matrix as coefficient matrix. *Opuscula Mathematica*, 2:33–45, 1986.

[24] J. Garloff. Block methods for the solution of linear interval equations. *SIAM J. Matrix Anal. Appl.*, 11:89–106, 1990.

[25] A. George, J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Englewood Cliffs, 1981.

[26] E. R. Hansen. Gaussian elimination in interval systems. *Preprint*. 1997.

[27] E. R. Hansen. The hull of preconditioned interval linear equations. *Reliable Computing*, 6(2):95–103, 2000.

[28] E. R. Hansen. Sharpening interval computations. *Reliable Computing*, 12(1):21–34, 2006.

[29] E. R. Hansen, R. Smith. Interval arithmetic in matrix computations, Part II. *SIAM J. Numer. Anal.*, 4:1–9, 1967.

[30] M. Hebgen. Eine scaling–invariante Pivotsuche für Intervallmatrizen. *Computing*, 12:99–106, 1974.

[31] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.

[32] T. Maier. *Intervall–Input–Output–Rechnung*. Mathematical Systems in Economics, 101, Hain, Königstein/Ts., 1985.

[33] G. Mayer. Enclosing the solution set of linear systems with inaccurate data by iterative methods based on incomplete LU–decomposition. *Computing*, 35:189–206, 1985.

[34] G. Mayer. Enclosing the solutions of systems of linear equations by interval iterative processes. *Computing Suppl.*, 6:47–58, 1988.

[35] G. Mayer. Old and new aspects of the interval Gaussian algorithm. In *Computer Arithmetic, Scientific Computation and Mathematical Modelling*. Eds.: E. Kaucher, S. M. Markov, G. Mayer. IMACS Annals on Computing and Applied Mathematics 12, Baltzer, Basel, 1991, pp. 329–349.

[36] G. Mayer. A contribution of the feasibility of the interval Gaussian algorithm. *Reliable Computing*, 12(2):79–98, 2006.

[37] G. Mayer, L. Pieper. A necessary and sufficient criterion to guarantee the feasibility of the interval Gaussian algorithm for a class of matrices. *Appl. Math.*, 38(3):205–220, 1993.

[38] G. Mayer, J. Rohn. On the applicability of the interval Gaussian algorithm. *Reliable Computing*, 4(3):205–222, 1998.

[39] J. Mayer. An approach to overcome division by zero in the interval Gaussian algorithm. *Reliable Computing*, 8(3):229–237, 2002.

[40] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, N.J., 1966.

[41] J. J. Moré. Nonlinear generalizations of matrix diagonal dominance with applications to Gauss–Seidel iterations. *SIAM J. Numer. Anal.*, 9:357–378, 1972.

[42] A. Neumaier. New techniques for the analysis of linear interval equations. *Linear Algebra Appl.*, 58:273–325, 1984.

[43] A. Neumaier. Further results on linear interval equations. *Linear Algebra Appl.*, 87:155–179, 1987.

[44] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[45] K. Nickel. Die Überschätzung des Wertebereichs einer Funktion in der Intervallrechnung mit Anwendungen auf lineare Gleichungssysteme. *Computing*, 18:15–36, 1977.

[46] K. Nickel. Interval–Analysis. In *The State of the Art in Numerical Analysis*. Ed.: D. A. H. Jacobs. Academic Press, London, 1977, pp. 193–225.

[47] J. M. Ortega. *Numerical Analysis. A Second Course*. Classics in Applied Mathematics 3, SIAM, Philadelphia, 1990.

[48] K. Reichmann. Ein hinreichendes Kriterium für die Durchführbarkeit des Intervall–Gauß–Algorithmus bei Intervall–Hessenberg–Matrizen ohne Pivotsuche. *Z. Angew. Math. Mech.*, 59:373–379, 1979.

[49] K. Reichmann. Abbruch beim Intervall–Gauss–Algorithmus. *Computing*, 22:355–361, 1979.

[50] K. Reichmann. Abbruch beim Intervall–Gauss–Algorithmus. *Freiburger Intervall–Berichte*, 79/5:1–30, 1979.

[51] J. Rohn. On overestimations produced by the interval Gaussian algorithm. *Reliable Computing*, 3(4):363–368, 1997.

[52] S. M. Rump. Verified solution of large systems and global optimization problems. *J. Comp. Appl. Math.*, 60:201–218, 1995.

[53] U. Schäfer. *Das lineare Komplementaritätsproblem mit Intervalleinträgen*. Thesis, Universität Karlsruhe, Karlsruhe, 1999.

[54] U. Schäfer. The feasibility of the interval Gaussian algorithm for arrowhead matrices. *Reliable Computing*, 7(4):59–62, 2001.

[55] U. Schäfer. Two ways to extend the Cholesky decomposition to block matrices with interval entries. *Reliable Computing*, 8(1):1–20, 2002.

[56] F. Schätzle. Überschätzung beim Gauss–Algorithmus für lineare Intervallgleichungssysteme. *Freiburger Intervall–Berichte*, 84/3:1–122, 1984.

[57] H. Schwandt. An interval arithmetic approach for the construction of an almost globally convergent method for the solution of nonlinear Poisson equation on the unit square. *SIAM J. Sci. Statist. Comput.*, 5:427–452, 1984.

[58] H. Schwandt. The solution of nonlinear elliptic Dirichlet problems on rectangles by almost globally convergent interval methods. *SIAM J. Sci. Statist. Comput.*, 6:617–638, 1985.

[59] H. Schwandt. Cyclic reduction for tridiagonal systems of equations with interval coefficients on vector computers. *SIAM J. Numer. Anal.*, 26:661–680, 1989.

[60] P. Wißkirchen. Ein Steuerungsprinzip der Intervallrechnung und dessen Anwendung auf den Gaußschen Algorithmus. *GMD–Bericht No. 20*, Bonn, 1969.

[61] P. Wongwises. *Experimentelle Untersuchungen zur numerischen Auflösung von linearen Gleichungssystemen mit Fehlererfassung*. Thesis, Universität Karlsruhe, Karlsruhe, 1974. Institut für Praktische Mathematik, Interner Bericht 75/1, 1975.

[62] P. Wongwises. Experimentelle Untersuchungen zur numerischen Auflösung von linearen Gleichungssystemen mit Fehlererfassung. In *Interval Mathematics*. Eds.: G. Goos, J. Hartmanis. Lecture Notes in Computer Science 29, Springer, Berlin, 1975, pp. 316–325.

Numerical Computation of the Mapping Degree using Computational Homology

Kansaku Nakakura
Graduate School of Mathematical Sciences
The University of Tokyo
3-8-1 Komaba, Meguro, Tokyo, 153-8914 Japan
kansaku@qf7.so-net.ne.jp

Sunao Murashige
Department of Complex Systems
School of Systems Information Science
Future University-Hakodate
116-2 Kamedanakano, Hakodate, Hokkaido, 041-8655 Japan
murasige@fun.ac.jp

Abstract

This paper describes numerical computation of the mapping degree $\deg(f, B^d)$ *for a continuous map* $f : B^d \to \mathbb{R}^d$ *on the* d*-dimensional ball* B^d *where* $d \in \mathbb{Z}$ *and* ≥ 2*. The mapping degree can be defined using a homomorphism which is induced on homology groups. We propose an efficient method to compute the homomorphism without direct calculation of homology groups, and obtain the mapping degree using computational homology.*

1. Introduction

The mapping degree $\deg(f, B^d)$ is an integer index defined for a continuous map $f : B^d \to \mathbb{R}^d$ on the d-dimensional ball B^d where $d \in \mathbb{Z}$ and ≥ 2. The mapping degree plays an important role in functional analysis, topology, and their applications to systematically understand some typical properties of a map, for example, existence and number of equilibrium points, bifurcation, complexity of dynamical system, and so on [1, 2, 3]. In particular, Kronecker's existence principle (for example, see [2], p.520), namely

$$\deg(f, B^d) \neq 0 \;\Rightarrow\; \exists x^* \in B^d \;:\; f(x^*) = 0\,, \qquad (1)$$

can be directly applied to numerical verification of solutions of $f(x) = 0$.

Since it is not straightforward to obtain the mapping degree for a general map, it is significant to develop a numerical method to compute the degree [4, 5, 6, 7, 8]. Most of previous works on numerical calculation of the mapping degree are based on analytical definition of the degree. On the other hand, it is also possible to define the mapping degree using homology groups as shown in section 2. Homological definition of the degree is irrelevant to smoothness of the map f or regularity of the Jacobi matrix $\partial f/\partial x$ in B^d, and the computational methods of the homology group have been developed recently [9].

This work tries to compute the mapping degree with homological definition using computational homology.

2. Homological definition of the mapping degree

The mapping degree $\deg(f, B^d) \in \mathbb{Z}$ for a continuous map $f : B^d \to \mathbb{R}^d$ $(0 \notin f(\partial B^d))$ on the d-dimensional ball B^d can be defined using the well-known relation of the homology group

$$H_{d-1}(S^{d-1}) \cong \mathbb{Z} \quad \text{for } d \in Z \text{ and } \geq 2\,, \qquad (2)$$

where H_{d-1} and S^{d-1} denote the $d-1$-dimensional homology group and sphere, respectively. For this definition, set $B^d = [-m, m]^d$ with a fixed positive integer m, $S^{d-1} = \partial([-1, 1]^d)$, and

$$\|x\|_{\max} := \max_{1 \leq i \leq d} |x_i|\,, \;\; x = (x_i) \in \mathbb{R}^d\,. \qquad (3)$$

Then we can define $\bar{f} : S^{d-1} \to S^{d-1}$ as

$$\bar{f}(x) := \frac{f(mx)}{\|f(mx)\|_{\max}} . \tag{4}$$

From the relation (2), we can fix $u \in H_{d-1}(S^{d-1})$ such that $\mathbb{Z}u = H_{d-1}(S^{d-1})$. Then, the induced homomorphism

$$\bar{f}_{*d-1} : H_{d-1}(S^{d-1}) \to H_{d-1}(S^{d-1}) , \tag{5}$$

uniquely determines an integer α such that

$$\bar{f}_{*d-1}(u) = \alpha u , \quad u \in H_{d-1}(S^{d-1}) . \tag{6}$$

This integer α is independent of choice of u. Then the mapping degree is defined by

$$\deg(f, B^d) := \alpha . \tag{7}$$

This work shows a method to calculate the induced homomorphism $\bar{f}_{*d-1}$ using computational homology, and obtain $\deg(f, B^d)$.

3. Computation of an induced homomorphism f_* using computational homology

Computational homology [9] enables us to calculate an induced homomorphism $f_* : H_*(X) \to H_*(Y)$ between homology groups $H_*(X)$ and $H_*(Y)$ for a continuous map $f : X \to Y$. Some terminologies used in computational homology are summarized in the appendix. The computational method of f_* is composed of the following four steps:

Step 1. Set a continuous map $f : X \to Y$ on cubical sets X and Y.

Set the domain X and the range Y of a continuous map $f : X \to Y$ such that these are 'cubical sets' given by Definition A.1 in the appendix. The homology group $H(X)$ of a cubical set X can be efficiently computed using the algorithm 'collapse' which reduces unnecessary bases of chain groups with the corresponding homology group kept [10, 11]. The computational complexity of this algorithm is $\mathrm{O}(n^3)$ where $n = |\mathcal{K}(X)|$ is the size of the chain group and $\mathcal{K}(X)$ denotes a set of all elementary sets included in a cubical set X, as shown in Definition A.1.

Step 2. Compute the multivalued map $F : X \to 2^Y$.

The next step is to determine the multivalued map $F : X \to 2^Y$ for a continuous map $f : X \to Y$ such that the following conditions are satisfied:

(i) For $\forall x \in X$, $f(x) \in F(x)$.

(ii) For $\forall x \in X$, $F(x)$ is a cubical set.

(iii) For $Q \in \mathcal{K}(X)$ and $\forall x, y \in \mathring{Q}$, $F(x) = F(y)$.

(iv) For $P, Q \in \mathcal{K}(X)$, $P \subset Q \;\Rightarrow\; F(\mathring{P}) \subset F(\mathring{Q})$.

(v) F is acyclic.

where $\mathring{Q}$ for $Q \in \mathcal{K}(X)$ is given in Definition A.1, and an acyclic multivalued map $F : X \to 2^Y$ is defined using a homology group as follows:

Definition 3.1 *(acyclic multivalued map)*
The multivalued map $F : X \to 2^Y$ is acyclic if, for $\forall x \in X$,

$$H_d(F(x)) \cong \begin{cases} \mathbb{Z} & \text{for } d = 0 , \\ 0 & \text{for } d \neq 0 . \end{cases} \tag{8}$$

□

Conditions (ii) and (iv) are required for preservation of topological properties of the original map f, (iii) is necessary for numerical examination of the acyclic condition as shown in step 3, and (iv) and (v) guarantee that F uniquely determines the chain homomorphism $\varphi : C(X) \to C(Y)$ such that $f_* = \varphi_* : H_*(X) \to H_*(Y)$. Here $C(X)$ denotes a set of chain groups of X.

It is straightforward to construct the multivalued map F, which satisfies conditions (i), (ii), (iii) and (iv), using the interval arithmetic. For $X \subset \mathbb{R}^d$, $Y \subset \mathbb{R}^{d'}$ and a continuous map $f : X \to Y$, such a multivalued map $F : X \to 2^Y$ is given by

$$F(x) = Y \cap \left(\bigcap \{ \check{f}(Q) \mid x \in Q \in \mathcal{K}(X) \} \right) , \tag{9}$$

where $Q = I_1 \times \cdots \times I_d$, $\check{f} := (\check{f}_1, \ldots, \check{f}_{d'})$ denotes a map obtained using the interval arithmetic for f, and

$$\begin{aligned} f(Q) &= f(I_1, \ldots, I_d) \\ &= (f_1(I_1, \ldots, I_d), \ldots, f_{d'}(I_1, \ldots, I_d)) . \end{aligned} \tag{10}$$

Here $\check{f}_i$ is computed such that its image is an elementary interval. The computational complexity for construction of

the multivalued map F is $\mathrm{O}(n^2 + nm)$ where $n = |\mathcal{K}(X)|$ and $m = |\mathcal{K}(Y)|$.

Step 3. Examine the acyclic condition for the multivalued map F.

Condition (v) in step 2, namely the acyclic condition for the multivalued map F, is satisfied with sufficiently fine division of the domain X. As shown in Definition 3.1, we can examine this condition by directly computing the homology group of $F(x)$ for $\forall x \in X$. Here it should be noted that this examination is enough for finite number of $x \in X$, not for $\forall x \in X$, because condition (iii) in step 2 is satisfied.

The computational complexity of the acyclic condition for F is at least $\mathrm{O}(nm^3)$ where $n = |\mathcal{K}(X)|$ and $m = |\mathcal{K}(Y)|$. Most of the computing time for f_* is used for construction of the multivalued map F and examination of the acyclic condition of F.

Step 4. Construct the chain homomorphism φ and obtain $f_* = \varphi_*$.

If the multivalued map F satisfies conditions (i)$\sim$(v), we can inductively construct the chain homomorphism $\varphi : C(X) \to C(Y)$ such that

$$\varphi_{k-1} \circ \partial_k = \partial_k \circ \varphi_k \ , \tag{11}$$

and

$$\left|\varphi_k(\widehat{Q})\right| \subset F(\mathring{Q}) \quad \text{for } \forall Q \in \mathcal{K}_k(X) \ . \tag{12}$$

First, set $\varphi_k = 0$ for $k < 0$, and, in the case of $k = 0$, arbitrarily choose $P \in \mathcal{K}_0(F(Q))$ for $\forall Q \in \mathcal{K}_0(X)$ and set

$$\varphi_0(\widehat{Q}) := \widehat{P} \ . \tag{13}$$

Next, for $k \geq 1$, calculate $c \in C_k(F(\mathring{Q}))$ such that

$$\partial_k(c) = \varphi_{k-1} \circ \partial_k(\widehat{Q}) \ , \tag{14}$$

and obtain φ_k by

$$\varphi_k(\widehat{Q}) := c \ . \tag{15}$$

These chain homomorphisms $\varphi_0, \varphi_1, \cdots, \varphi_d$ determine the induced homomorphism f_*.

Next section applies this method to computation of the mapping degree.

4. Computation of the mapping degree using computational homology

4.1. A method to obtain the mapping degree from a chain homomorphism φ

For a continuous map $f : B^d \to \mathbb{R}^d$, we can set $\bar{f} : S^{d-1} \to S^{d-1}$ as (4), and calculate the corresponding chain homomorphism $\varphi : C(S^{d-1}) \to C(S^{d-1})$ through steps 1$\sim$4 in section 3 using computational homology. Then φ_{d-1} determines the mapping degree $\deg(f, B^d)$ as follows:

A base v of $H_{d-1}(S^{d-1}) \cong \mathbb{Z}$ is given by

$$v \ = \ \partial_d \left(\sum_{l_1,\ldots,l_d=\pm 1} \widehat{Q}_{l_1,\ldots,l_d} \right) \ , \tag{16}$$

where $Q_{l_1,\ldots,l_d} = [l_1, l_1+1] \times \cdots \times [l_d, l_d+1] \in \mathcal{K}^d$, and $\widehat{Q} : \mathcal{K}^d \to \mathbb{Z}$ and the cubical boundary map ∂_d are defined in Definition A.2 and A.3, respectively. Using this base v, we can determine the mapping degree $\deg(f, B^d) = \alpha \in \mathbb{Z}$ by

$$\varphi_{d-1}(v) \ = \ \alpha v \ . \tag{17}$$

4.2. An example to numerically obtain the mapping degree

As an example, consider the map $f : B^2 = [-1, 1]^2 \to \mathbb{R}^2$ defined by

$$f(x_1, x_2) \ = \ ({x_1}^2 + 2x_1 + 4x_2 \ , \ {x_2}^2 + 3x_2) \ . \tag{18}$$

Figures 1 (a) and (b) show the corresponding map $\overline{f} : S^1 \to S^1$ where $S^1 = \partial B^2 = \partial[-1, 1]^2$ where both the domain S^1 and the range S^1 are divided into eight elements. In figs.1 (b), (c) and (d), the axis of abscissa denotes the domain S^1 and that of ordinate the range S^1, respectively. Figure 1 (c) depicts the multivalued map F computed using the interval arithmetic for $\bar{f}$. For this multivalued map F in fig.1 (c), the acyclic condition is not satisfied on the element at the right end of the domain. Then the domain S^1 is subdivided into finer elements so that the acyclic condition is satisfied on all elements as shown in fig.1 (d), in which the corresponding chain homomorphism φ_1 is also drawn by the broken line. Using (16), (17) and this φ_1, we can obtain

$$
\begin{aligned}
&\varphi_1([0,1]+[1,2]+\cdots+[15,16]) \\
&= [0,1]-[0,1]+([0,1]+[1,2]+[2,3]+[3,4]) \\
&\quad +[4,5]+([5,6]+[6,7])+[7,8] \\
&= [0,1]+[1,2]+\cdots+[7,8]\,.
\end{aligned} \tag{19}
$$

Thus the mapping degree $\deg(f, B^d) = 1$.

4.3. An efficient method to examine the acyclic condition for a map $\bar{f} : S^d \to S^d$

As stated in section 3, it takes much computing time to examine the acyclic condition using the homology group. This subsection proposes an efficient method to check the acyclic condition for a map $\bar{f} : S^d \to S^d$ given by (4) without direct computation of the homology group.

The induced homomorphism $\bar{f}_* : H_*(S^d) \to H_*(S^d)$ on the homology group can be computed using the multivalued map

$$
F(x) = S^d \cap \left(\bigcap \{ \check{f}(Q) \mid x \in Q \in \mathcal{K}(S^d) \} \right) \tag{20}
$$

where $\check{f}$ denotes a map obtained using the interval arithmetic for f, as shown in step 2 of section 3. Here, since $\check{f}(Q)$ is a rectangle as shown in fig.2, $\bigcap\{\check{f}(Q) \mid x \in Q \in \mathcal{K}(S^d)\}$ is also a rectangle. Thus we need to examine the acyclic condition for $R \cap S^d (\neq \emptyset)$ where R is a rectangle. For that, fix integers $c_1^{\pm}, c_2^{\pm}, \ldots, c_d^{\pm}$ $(c_i^- < c_i^+)$, set

$$
X = [c_1^-,\ c_1^+] \times [c_2^-,\ c_2^+] \times \cdots \times [c_d^-,\ c_d^+] \tag{21}
$$

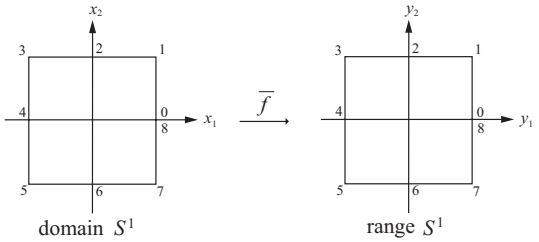
and consider $R \cap \partial X (\neq \emptyset)$ where the boundary ∂X is given by

$$
\partial X = \bigcup_{i=1}^{d} (X_i^+ \cup X_i^-) \tag{22}
$$

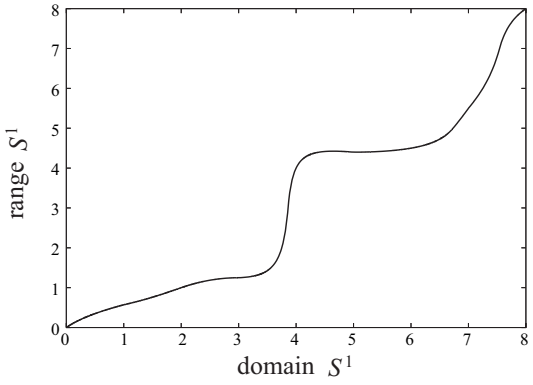
with

$$
\begin{aligned}
X_i^+ &= \left(\prod_{j=1}^{i-1} [c_j^-,\ c_j^+] \right) \times [c_i^+,\ c_i^+] \times \left(\prod_{j=i+1}^{d} [c_j^-,\ c_j^+] \right), \\
X_i^- &= \left(\prod_{j=1}^{i-1} [c_j^-,\ c_j^+] \right) \times [c_i^-,\ c_i^-] \times \left(\prod_{j=i+1}^{d} [c_j^-,\ c_j^+] \right).
\end{aligned}
$$

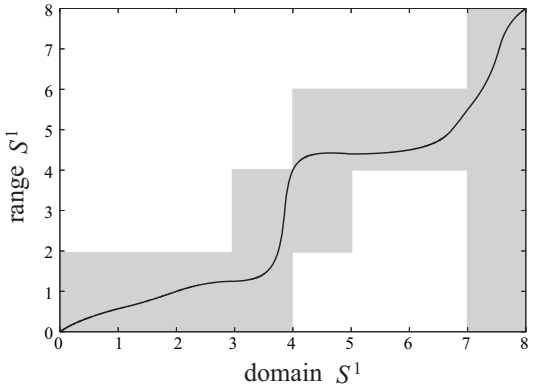
Also $R \subset \mathbb{R}^d$ can be expressed as



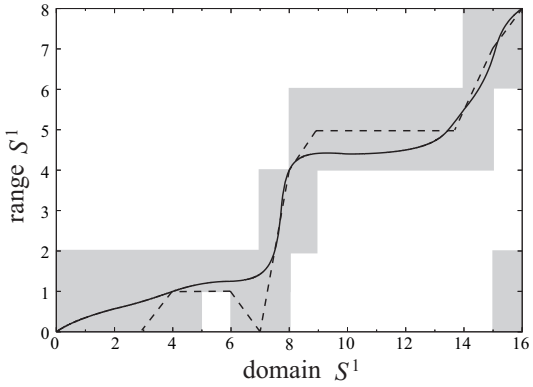
(a) The domain and the range of $\bar{f}$



(b) The map $\bar{f} : S^1 \to S^1$



(c) The multivalued map F
(solid : $\bar{f}$, gray : the image of F)



(d) The multivalued map F with fine subdivision of the domain (solid : $\bar{f}$, gray : the image of F, broken : the chain homomorphism φ_1)

Figure 1. Computation of the multivalued map F for the map f in (18)

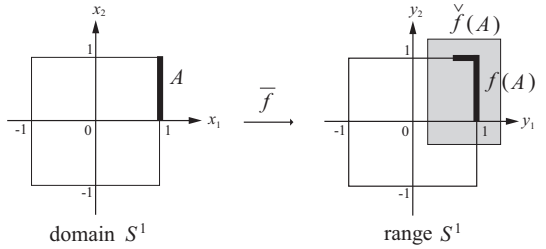


Figure 2. The image of $\breve{f}$ in (20)

$$R = [a_1,\ b_1] \times \cdots \times [a_d,\ b_d] \qquad (a_i, b_i \in \mathbb{Z})\ . \tag{23}$$

Since $R \cap \partial X = (R \cap X) \cap \partial X$, we can assume that $c_i^- \leq a_i,\ b_i \leq c_i^+$. Then we can get the following proposition:

Proposition 4.1 *$\partial X \cap R$ is not acyclic if and only if, for $i = 1, \ldots, d$, $a_i = c_i^- \Leftrightarrow b_i = c_i^+$.*

Proof. (Necessity) We show the contraposition of the necessity. There exists i such that $a_i = c_i^-,\ b_i \neq c_i^+$ or $a_i \neq c_i^-,\ b_i = c_i^+$. Here suppose $a_i \neq c_i^-,\ b_i = c_i^+$. With suitable change of numbering, we can set $a_1 \neq c_1^-,\ b_1 = c_1^+$. Define $\varphi : \partial X \cap R \to \partial X_1^+ \cap R$ as

$$\varphi(x_1, \ldots, x_d) := (c_1^+, x_2, \ldots, x_d)\ , \tag{24}$$

and let $\psi : \partial X_1^+ \cap R \hookrightarrow \partial X \cap R$ be an inclusion map. Since $\partial X_1^+ \cap R$ is acyclic, the necessary condition is satisfied if $\varphi \circ \psi \simeq 1_{\partial X_1^+ \cap R}$ and $\psi \circ \varphi \simeq 1_{\partial X \cap R}$. Furthermore, since $\varphi \circ \psi = 1_{\partial X_1^+ \cap R}$, it is enough to show $\psi \circ \varphi \simeq 1_{\partial X \cap R}$. For that, define $F : (\partial X \cap R) \times [0, 1] \to \partial X \cap R$ as

$$F(x_1, \ldots, x_d, t) := ((1-t)x_1 + tc_1^+, x_2, \ldots, x_d)\ , \tag{25}$$

and show $\operatorname{im} F \subset \partial X \cap R$. For $(x, t) = (x_1, \ldots, x_d, t) \in (\partial X \cap R) \times [0, 1]$, $F(x, t) \in R$ because

$$a_1 \leq x_1 \leq (1-t)x_1 + tc_1^+ \leq c_1^+ = b_1\ . \tag{26}$$

On the other hand, since $x \in \partial X$, there exists i such that $x \in X_i^+ \cup X_i^-$. In the case of $i = 1$, $x \in X_1^+$ because $c_1^- < a_1 \leq x_1$. Thus $x_1 = c_1^+$, and $F(x, t) = (c_1^+, x_2, \ldots, x_d) \in X_1^+$. In the case of $i > 1$, $x_i = c_i^\pm$, and so $F(x, t) = ((1-t)x_1 + tc_1^+, x_2, \ldots, c_i^\pm, \ldots, x_d) \in X_i^\pm$. From these, $F(x, t) \in \partial X$. Since F is a homotopy between $\psi \circ \varphi$ and $1_{\partial X \cap R}$, we can get $\psi \circ \varphi \simeq 1_{\partial X \cap R}$.

(Sufficiency) With suitable change of numbering, we can express R as

$$R = \left(\prod_{i=1}^{r} [c_i^-,\ c_i^+] \right) \times \left(\prod_{i=r+1}^{d} [a_i,\ b_i] \right), \tag{27}$$

where $(c_i^- < a_i \leq b_i < c_i^+,\ i = r+1, \ldots, d)$. Here $r \geq 1$ because $\partial X \cap R \neq \emptyset$. Set X' as

$$X' = [c_1^-,\ c_1^+] \times \cdots \times [c_r^-,\ c_r^+]\ , \tag{28}$$

and define $\varphi : \partial X \cap R \to \partial X'$ by

$$\varphi(x_1, \ldots, x_d) := (x_1, \ldots, x_r)\ , \tag{29}$$

and $\psi : \partial X' \to \partial X \cap R$ by

$$\psi(x_1, \ldots, x_r) := (x_1, \ldots, x_r, a_{r+1}, \ldots, a_d)\ , \tag{30}$$

respectively. First we can show that $\operatorname{im} \varphi \subset \partial X'$ and $\operatorname{im} \psi \subset \partial X \cap R$. When $x = (x_1, \ldots, x_d) \in \partial X \cap R$, $x \in \partial X$ and there exists i such that $x_i = c_i^\pm$. Since $x \in R$, $i \leq r$. Thus $\operatorname{im} \varphi \subset \partial X'$. When $x = (x_1, \ldots, x_r) \in \partial X'$, there exists $i \leq r$ such that $x_i = c_i^\pm$, and so $\psi(x) \in \partial X$. Since $\psi(x) \in R$, $\operatorname{im} \psi \subset \partial X \cap R$.

$\partial X'$ is not acyclic because it is homeomorphic to S^{r-1}. Thus we just need to show $\varphi \circ \psi \simeq 1_{\partial X'},\ \psi \circ \varphi \simeq 1_{\partial X \cap R}$. Since $\varphi \circ \psi = 1_{\partial X'}$, it is enough to show $\psi \circ \varphi \simeq 1_{\partial X \cap R}$. Define $F : (\partial X \cap R) \times [0, 1] \to \partial X \cap R$ as

$$\begin{aligned} &F(x_1, \ldots, x_d, t) := \\ &(x_1, \ldots, x_r, ta_{r+1} + (1-t)x_{r+1}, \ldots, ta_d + (1-t)x_d)\ . \end{aligned} \tag{31}$$

We can show $\operatorname{im} F \subset \partial X \cap R$. When $(x, t) = (x_1, \ldots, x_d, t) \in (\partial X \cap R) \times [0, 1]$, $x \in \partial X$ and so there exists i such that $x_i = c_i^\pm$. Since $x \in R$, $i \leq r$. From these, $F(x, t) \in \partial X$. In addition,

$$a_{r+j} \leq ta_{r+j} + (1-t)x_{r+j} \leq x_{r+j} \leq b_{r+j}\ , \tag{32}$$

and so we can get $F(x, t) \in R$. Since F is a homotopy between $1_{\partial X \cap R}$ and $\psi \circ \varphi$, $\psi \circ \varphi \simeq 1_{\partial X \cap R}$. □

This proposition shows that we can examine the acyclic condition of $F(x)$ with the computational complexity $\mathrm{O}(d)$. On the other hand, direct computation of the homology group for the acyclic condition requires much more computational cost. For example, the worst case is $F(x) = S^d$ for which

$$|\mathcal{K}(S^d)| = 5^{d+1} - 3^{d+1} \tag{33}$$

and the computational complexity only for 'collapse' used in step 1 of section 3 is $\mathrm{O}(125^d)$.

5. Numerical examples

5.1. Example 1 (non-singular case)

The first example is the inclusion map $f : B^d \hookrightarrow \mathbb{R}^d$ on the domain $B^d = [-1, 1]^d$ given by

$$f(x) \;=\; x \;, \quad x \in B^d \;. \tag{34}$$

The mapping degree for this is $\deg(f, B^d) = 1$ and could be obtained using the proposed method. The computing time is shown in fig.3, in which 'method 1' and 'method 2' examine the acyclic condition (8) with and without direct calculation of the homology groups, respectively. Method 2 corresponds to the proposed method using Proposition 4.1. This result suggests that the proposed method efficiently reduces the computing time with increase of the dimension d.

In order to obtain the chain homomorphism φ_d, this method requires computation of $\varphi_0, \varphi_1, \cdots, \varphi_{d-1}$, as shown in step 4 of section 3. This leads to much computational cost for a system of large dimension. For this example (34), we could not obtain the degree for the dimension $d \geq 6$ due to the memory size. This problem remains as a future work.

5.2. Example 2 (singular at the center of a domain B)

The second example is the map $f : B^2 = [-1, 1]^2 (\subset \mathbb{C}) \to \mathbb{C}$ given by

$$f(z) = z^n \qquad (n = 0, 1, \cdots) \;, \tag{35}$$

where $z = x_1 + \mathrm{i}\, x_2 \in \mathbb{C}$ and $(x_1, x_2) \in \mathbb{R}^2$ is identified with $z \in \mathbb{C}$. For example, in the case of $n = 2$,

$$f(x_1, x_2) \;=\; (x_1^2 - x_2^2 \;,\; 2x_1 x_2) \;. \tag{36}$$

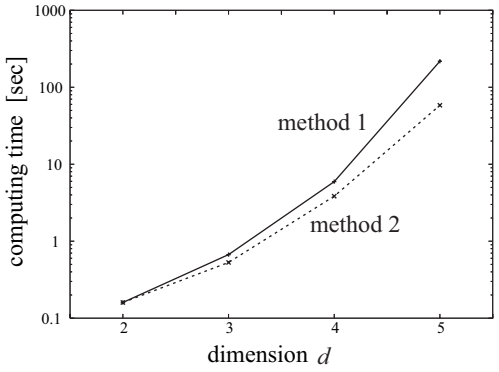


Figure 3. The computing time for example 1, the inclusion map (34). solid (method 1) : the acyclic condition (8) is examined with direct calculation of the homology group. broken (method 2) : (8) is examined using the proposed method.

The mapping degree of this map is $\deg(f, B^2) = n$, and the Jacobi matrix $\partial f/\partial x$ is singular at the origin $z = 0$. Figure 4 shows the computing time for different degree n. It is found that the computing time increases with the degree n, because more subdivision of the domain is required. Also, since the dimension $d = 2$, the computing time by method 1 is almost the same as that by method 2.

This is just an illustrative example to show that the proposed method can be applied to this type of singular case. Aberth's method can also produce the degree for this case [6, 8].

5.3. Example 3 (singular at a point on the boundary ∂B)

The last example is the map $f : B_c^2 \to \mathbb{R}^2$ given by

$$f(x_1, x_2) \\ = \; \left(\tfrac{1}{10} x_1 + \tfrac{1}{100} {x_2}^2 + c^2 \;,\; \tfrac{1}{100} x_1 x_2 + \tfrac{1}{10} x_2\right) \;. \tag{37}$$

This corresponds to a two-dimensional case of a mathematical model of water waves progressing into one direction with a constant speed c in water of infinite depth [8, 12]. Here the domain B_c^2 $(\subset \mathbb{R}^2)$ defined by

$$B_c^2 := [-1 - 10c^2, 1 - 10c^2] \times [-1, 1] \;, \tag{38}$$

is the rectangle of which the center x_c given by

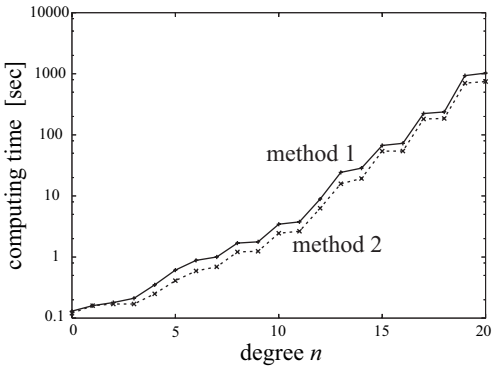


Figure 4. The computing time for example 2, (35). See caption in fig.3.

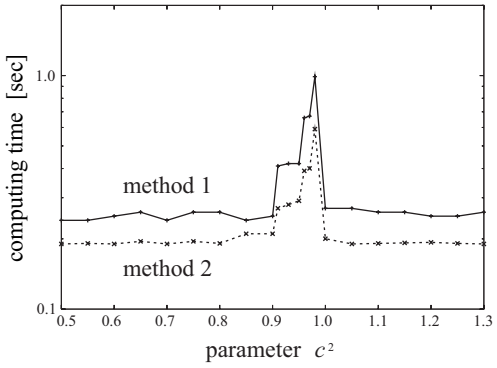


Figure 5. The computing time for example 3, (37). See caption in fig.3.

$$x_c = (-10c^2,\ 0)\ , \tag{39}$$

is located at one of solutions of $f(x) = 0$. Then the mapping degree $\deg(f, B_c^2)$ is given by

$$\deg(f, B_c^2) \ = \ \begin{cases} 1 & (0 \le c^2 < 0.99)\ , \\ -1 & (0.99 < c^2)\ . \end{cases} \tag{40}$$

At $c^2 = 0.99$, $0 \in f(\partial B_c^2)$ and the degree cannot be defined. Also it should be noted that the Jacobi matrix $\partial f/\partial x$ is singular at a point on the boundary ∂B_c^2 for $0.88 \le c^2 \le 1.1$.

Figure 5 compares the computing time for different values of the parameter c^2. It is found that the computing time using the proposed method 'method 2' is less than that using 'method 1', although the difference is small. Also it should be noted that the computing time increases for $0.9 < c^2 < 1$. It is because there exist two other solutions $x_a = (-10,\ 10\sqrt{1-c^2})$ and $x_b = (-10,\ -10\sqrt{1-c^2})$ near the boundary ∂B_c^2, the denominator $\|f\|_{\max}$ of $\bar{f}$ defined by (4) is close to zero at $x \simeq x_a$ or x_b, and the interval arithmetic overestimates the corresponding intervals. For $1 < c^2 < 1.1$, there exist only one solution x_c and the computing time remains small even when the Jacobi matrix $\partial f/\partial x$ is singular at a point on the boundary ∂B_c^2.

These examples show that the proposed method can be applied even when the Jacobi matrix $\partial f/\partial x$ is singular at a point in a domain B or on its boundary ∂B.

6. Conclusions

This work has considered numerical computation of the mapping degree $\deg(f, B^d)$ for a continuous map $f : B^d \to \mathbb{R}^d$ on the d-dimensional ball B^d using computational homology. The degree can be determined using the corresponding map $\overline{f} = f/\|f\| : S^{d-1} \to S^{d-1}$ and the induced homomorphism $\overline{f}_* : H_*(S^{d-1}) \to H_*(S^{d-1})$ on the homology group $H_*(S^{d-1})$. In order to reduce the computational cost, we have proposed the method to obtain the degree without direct calculation of homology groups. Numerical examples using simple maps indicate that the proposed method works even for singular cases.

For practical application of this method, the problem of computational complexity still remains for a system of large dimension. In order to overcome this problem, we plan to reconsider construction of the chain homomorphism φ in future works.

References

[1] J. Cronin, *Fixed points and topological degree in nonlinear analysis*, Amer. Math. Soc., Providence, RI, 1964.

[2] E. Zeidler, *Nonlinear functional analysis and its applications I, Fixed-point theorems*, Springer, 1986.

[3] E.H. Rothe, *Introduction to various aspects of degree theory in Banach spaces*, American Mathematical Society, 1986.

[4] F. Stenger, "Computing the topological degree of a mapping in $\mathbb{R}^n$", *Numeri. Math.*, vol.25, pp.23–38, 1975.

[5] R.B. Kearfott, "An efficient degree-computation method for a generalized method of bisection", *Numeri. Math.*, vol.32, pp.109–127, 1979.

[6] O. Aberth, "Computation of topological degree using interval arithmetic, and applications", *Mathematics of Computations*, vol.62, no.205, pp.171–178, 1994.

[7] R.B. Kearfott, J. Dian and A. Neumaier, "Existence verification for singular zeros of complex nonlinear systems", *SIAM J. Numeri. Anal.*, vol.38, no.2, pp.360–379, 2000.

[8] S. Murashige, "A practical method of numerical calculation of the mapping degree", *IEICE Trans. Fundamentals*, vol.E89-A, no.6, pp.1813–1819, 2006.

[9] T. Kaczynski, K. Mischaikow and M.Mrozek, *Computational homology*, Applied Mathematical Sciences, vol.157, Springer-Verlag, New York, 2004.

[10] T. Kaczynski, M. Mrozek and M. Ślusarek, "Homology computation by reduction of chain complexes", *Comput. Math.*, vol.35, pp.59–70, 1998.

[11] W. Kalies, K. Mischaikow and G. Watson, "Cubical approximation and computation of homology", *Banach Center Publications*, vol.47, pp.115–131, 1999.

[12] M.S. Longuet-Higgins, "Bifurcation in gravity waves", *J. Fluid Mech.*, vol.151, pp.457–475, 1985.

A Some definitions in computational homology [9]

Definition A.1 *(cubical set)*

Define an elementary interval $I \subset \mathbb{R}$ by $I = [l, l+1]$ or $I = [l, l]$ ($l \in \mathbb{Z}$), and the associated elementary cell $\mathring{I}$ by

$$\mathring{I} := \begin{cases} [l,l] & \text{for } I = [l,l]\,, \\ (l,l+1) & \text{for } I = [l,l+1]\,. \end{cases} \tag{41}$$

Define an elementary cube $Q \subset \mathbb{R}^d$ and the associated elementary cell $\mathring{Q}$ by

$$\begin{aligned} Q &:= I_1 \times \cdots \times I_d \\ \mathring{Q} &:= \mathring{I}_1 \times \cdots \times \mathring{I}_d \end{aligned} \tag{42}$$

where $I_1, \ldots, I_d$ denotes elementary intervals. The dimension $\dim Q$ of $Q = I_1 \times \cdots \times I_d$ is defined by the number of elementary intervals such that $I_j = [l, l+1]$ ($l \in \mathbb{Z}$). If $X \subset \mathbb{R}^d$ can be written as a finite union of elementary cubes, then X is called a 'cubical set'.

Let $\mathcal{K}^d$ be a set of all elementary cubes in $\mathbb{R}^d$, and we use the following notations:

$$\begin{aligned} \mathcal{K}_k^d &:= \{Q \in \mathcal{K}^d \,:\, \dim Q = k\}\,, \\ \mathcal{K} &:= \bigcup_{d=1}^{\infty} \mathcal{K}^d\,, \\ \mathcal{K}(X) &:= \{Q \in \mathcal{K} \,:\, Q \subset X\}\,, \\ \mathcal{K}_k(X) &:= \{Q \in \mathcal{K}(X) \,:\, \dim Q = k\}\,, \end{aligned} \tag{43}$$

where $X \subset \mathbb{R}^d$ is a cubical set. Also, let $|\mathcal{K}(X)|$ be the number of elements of $\mathcal{K}(X)$.

□

Definition A.2 *(cubical chain group)*

The cubical chain group $C_k(X)$ of k-dimension for a cubical set $X \subset \mathbb{R}^d$ is defined by

$$C_k(X) := \{\alpha_1 \widehat{Q}_1 + \cdots + \alpha_n \widehat{Q}_n \,:\, \alpha_i \in \mathbb{Z}\,,\ Q_i \in \mathcal{K}_k(X)\}\,, \tag{44}$$

where $\widehat{Q} : \mathcal{K}_k^d \to \mathbb{Z}$ for $Q \in \mathcal{K}_k^d$ is defined by

$$\widehat{Q}(P) := \begin{cases} 1 & P = Q\,, \\ 0 & P \neq Q\,, \end{cases} \tag{45}$$

and $C_k(X) = 0$ for $k > d$ or < 0.

For $c \in C_k(X)$ given by

$$c = \alpha_1 \widehat{Q}_1 + \cdots + \alpha_n \widehat{Q}_n\,, \quad \alpha_i \in \mathbb{Z} \backslash \{0\}\,, \tag{46}$$

the support $|c|$ is defined by

$$|c| := \bigcup_{j=1}^{n} Q_j\,. \tag{47}$$

□

Definition A.3 *(cubical boundary map)*

The cubical boundary map $\partial_k : C_k(X) \to C_{k-1}(X)$ for a cubical set $X \subset \mathbb{R}^d$ is defined by

$$\partial_k(\widehat{Q}) := \sum_{j=1}^{k} (-1)^{j-1} (\widehat{Q}_j^+ - \widehat{Q}_j^-)\,, \tag{48}$$

where $Q = I_1 \times \cdots \times I_d \in \mathcal{K}_k(X)$,

$$
\begin{aligned}
Q_j^- &:= I_1 \times \cdots \times I_{i_j-1} \times [l_j, l_j] \times I_{i_j+1} \times \cdots \times I_d \,, \\
Q_j^+ &:= I_1 \times \cdots \times I_{i_j-1} \times [l_j+1, l_j+1] \times I_{i_j+1} \\
&\quad \times \cdots \times I_d \,,
\end{aligned}
\tag{49}
$$

and $I_{i_1}, \ldots, I_{i_k}$ are subsequences of $I_1, \ldots, I_d$ such that $I_{i_j} = [l_j, l_j+1]$. This definition can be linearly extended to a general $c \in C_k(X)$. □

Computer-Assisted Proofs in Solving Linear Parametric Problems

Evgenija D. Popova
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev str., block 8, 1113 Sofia, Bulgaria
epopova@bio.bas.bg

Abstract

Consider a linear system $A(p)x = b(p)$ whose input data depend on a number of uncertain parameters $p = (p_1,\ldots,p_k)$ varying within given intervals $[p]$. The objective is to verify by numerical computations monotonic (and convexity/concavity) dependence of a solution component $x_i(p)$ with respect to a parameter p_j over the interval box $[p]$, or more general, to prove if some boundary $\inf / \sup x_i(p)$ for all $p \in [p]$ is attained at the end-points of $[p]$. Such knowledge is useful in many applications in order to facilitate the solution of some underlying linear parametric problem involving uncertainties.

In this paper we present a technique, for proving the desired properties of the parametric solution, which is alternative to the approaches based on extreme point computations. The proposed computer-aided proof is based on guaranteed interval enclosures for the partial derivatives of the parametric solution for all $p \in [p]$. The availability of self-validated methods providing guaranteed enclosure of a parametric solution set by floating-point computations is a key for the efficiency and the expanded scope of applicability of the proposed approach. Linear systems involving nonlinear parameter dependencies, and dependencies between $A(p)$ and $b(p)$, as well as non-square linear parametric systems can be handled successfully. Presented are details of the algorithm design and Mathematica *tools implementing the proposed approach. Numerical examples from structural mechanics illustrate its application.*

1. Introduction

Consider the linear algebraic system

$$A(p)x = b(p), \tag{1}$$

where the elements of the $n \times n$ matrix $A(p)$ and the vector $b(p)$ depend on a k-tuple of parameters $p = (p_1,\ldots,p_k)$ which are uncertain and varying within given intervals

$$a_{ij}(p) = a_{ij}(p_1,\ldots,p_k), \quad b_i(p) = b_i(p_1,\ldots,p_k), \quad i,j = 1,\ldots,n, \tag{2}$$

$$p \in [p] = ([p_1],\ldots,[p_k]). \tag{3}$$

Such systems are common in many engineering analysis or design problems, models in operational research, linear prediction problems, etc., where there are complicated dependencies between the coefficients of the system [1]-[4], [7], [13]. The uncertainties in the model parameters could originate from an inexact knowledge of these parameters, measurement imprecision, or round-off errors.

In this paper a real compact interval is denoted by $[a] = [a^-, a^+] := \{a \in \mathbb{R} \mid a^- \le a \le a^+\}$. By $\mathbb{IR}^n, \mathbb{IR}^{n\times m}$ we denote the sets of interval n-vectors, resp. interval $n \times m$ matrices. For $[a] = [a^-, a^+]$, $0 \notin [a]$, define $\mathrm{sgn}([a]) := \{1 \text{ if } a^- > 0,\ -1 \text{ if } a^+ < 0\}$. The end-point functionals $(\cdot)^-, (\cdot)^+$ and the sgn functional are applied to interval vectors and matrices componentwise. We assume the reader is familiar with conventional interval arithmetic [6]. Let $U(k) := \{u \in \mathbb{R}^k \mid |u| = (1,...,1)^\top\}$, where the absolute value is understood componentwise, $|u| = (|u_1|,\ldots,|u_k|)^\top)$ for $u \in \mathbb{R}^k$. For $[a] \in \mathbb{IR}^n$ and $u \in U(n)$, a^u is defined by $a_i^u := \{a_i^- \text{ if } u_i = 1; a_i^+ \text{ if } u_i = -1\}$, $i = 1,\ldots,n$. For a set of indices $\mathcal{I} = \{i_1,\ldots,i_n\}$, the vector $(x_{i_1},\ldots,x_{i_n})^\top$ will be denoted by $x_{\mathcal{I}}$.

The set of solutions to (1–3), called parametric solution set, is

$$\begin{aligned}\Sigma^p &= \Sigma\,(A(p), b(p), [p]) \\ &:= \{x \in \mathbb{R}^n \mid \exists\, p \in [p], A(p)x = b(p)\}\,.\end{aligned} \tag{4}$$

In many applications, when the solution components $x_i = x_i(p)$ of (1–3) are proven to be convex, concave or monotonic, the solution of some underlying problem of interest can be facilitated. Most often, for a non-empty bounded set (4), one needs to find its convex hull

$$\square\,\Sigma^p := [\inf \Sigma^p,\ \sup \Sigma^p] = \cap\{[x] \in \mathbb{IR}^n \mid \Sigma^p \subseteq [x]\},$$

or an interval vector $[y] \supseteq \square\Sigma^p$ which is most tight. The following theorem is well-known.

Theorem 1 *Let $A(p)$ be nonsingular for all $p \in [p]$ and let the components of $x(p) = A^{-1}(p)b(p)$ be monotonic in $[p]$ with respect to each p_ν, $\nu = 1,\ldots,k$. Then for $i = 1,\ldots,n$*

$$\{\square\Sigma^p\}_i = \left[\left\{A^{-1}(p^u)\right\}_i b(p^u),\ \left\{A^{-1}(p^{-u})\right\}_i b(p^{-u})\right],$$

where $u_\nu = \text{sign}\frac{\partial x_i(p)}{\partial p_\nu}$, $\nu = 1,\ldots,k$.

Even when the solution components are not monotonic in $[p]$ monotonicity can also be used provided the behavior of the solution is such that the following property holds

$$\square\Sigma^p = [\min_{u\in U(k)} A^{-1}(p^u)b(p^u),\ \max_{u\in U(k)} A^{-1}(p^u)b(p^u)].$$

Such parametric solution sets will be called *possessing the combinatorial property*.

Practical examples exploiting the above properties are common in the design and modelling under deterministic uncertainties [1], [2], [13]. Similar examples can also be given in a stochastic context. For example, when it can be established that p enters a performance measure in a convex or concave manner, it becomes possible to carry out a so-called robust Monte Carlo simulation [2]. Also, when the components of p correspond to design parameters and $x_i(p)$ is some system quantity to be minimized, the convex dependence on p facilitates computation of some optimal setting $p = p^*$ for the system.

In order to be exploited, monotonicity, convexity/concavity, or combinatorial property of the parametric solution should be proven. Given a parametric system (1–3) with nonsingular matrix $A(p)$ for all $p \in [p]$, a solution component $x_i(p)$ is monotonic or convex/concave w.r.t. a parameter p_ν if $\frac{\partial x_i(p)}{\partial p_\nu}$, or $\frac{\partial^2 x_i(p)}{\partial p_\nu^2}$ respectively, have the same sign for all $p \in [p]$. For rank-one uncertainty structures[1], some extreme point results are given in [2]. Namely, it is proven that the desired same-sign conditions can be ascertain by checking the sign of certain multilinear affine functions at the (extreme) end-points of $[p]$. Another work [8] contains some verifiable sufficient conditions under which the parametric solution set (4) possesses the combinatorial property, or some bounds of the parametric solution set can be obtained by extreme point computations. The present work is motivated by the deficiencies of the approaches used in [2], [8]. First, proving solution set properties by extreme point computations is applicable only to linear systems with rank-one uncertainty structure. Second, point computations are guaranteed only if performed in exact arithmetic. Third, the computational efficiency of extreme point computations performed in exact arithmetic is too low.

[1] A pair $(A(p), b(p))$ has rank-one uncertainty structure if each parameter p_ν enters into either $A(p)$ or $b(p)$ but not both and the dependencies are affine-linear, cf. [2].

In view that proving the discussed properties of the parametric solutions is important also for other more general problem classes, e.g. such involving nonlinear parameter dependencies or non-square systems, in this work we present another approach for proving the desired properties. The computer-aided proof will be based on verifiable signs for the first partial derivatives (respectively the second partial derivatives for convexity/concavity property) of the parametric solution. The new line of attack consists in the way the proof will be done, namely, by self-validated solver of parametric interval linear systems which provides guaranteed solution enclosure in floating-point computations. In Section 2 the proposed alternative approach is presented in details together with its algorithmic design providing best computational efficiency. Some software tools implementing the proposed approach are presented in Section 3. This section demonstrates a much expanded scope of applicability of the proposed technique. Examples from structural mechanics illustrate its application. The computational efficiency of a proof based on self-validated parametric solver is compared to the approach based on extreme point computations.

2. Computer-aided proofs

The pioneering idea for computer-assisted monotonicity proofs related to parametric linear systems is due to Jiří Rohn [14]. He considered linear systems with rank-one uncertainty structure and employed the information about the monotonicity of the parametric solution with respect to the system parameters in order to obtain an enclosure of the parametric solution set. To prove the monotonicity properties Rohn's approach is based on enclosing the solution sets of corresponding linear systems for the partial derivatives of the original system. The deficiency of this first attempt in proving monotonicity properties of the parametric solution is that the linear systems for the partial derivatives are solved by "any classical method" for enclosing the solution of nonparametric interval linear systems. In other words, the idea of Rohn was to reduce the solution of a parametric linear system to the solution of several nonparametric interval linear systems and then to solve point linear systems corresponding to the monotonicity types. Since solving the derivative systems, which are also parametric, by nonparametric methods is practically useless due to the huge overestimation of the solutions (derivatives), Rohn's approach was improved in [9] by solving the derivative systems as parametric ones. In this section we further elaborate this approach ensuring its computational efficiency.

Consider the system of linear equations (1–3). Each parameter may enter both $A(p)$ and $b(p)$, and there is no restriction on the dependence between the parameters — $a_{ij}(p)$ and $b_i(p)$, $i,j = 1,\ldots,n$, could be either affine-

linear or nonlinear functions of the parameters. The latter are considered to vary within given intervals $[p_1], \ldots, [p_k]$. In the sequel we assume that $A(p)$ is nonsingular for all $p \in [p]$ and there is a function implementing some method for guaranteed enclosure of the solution set (4) of (1–3). Such methods exist, e.g. a general-purpose self-validating method for parametric linear systems is proposed in [16] and generalized in [10]. For its implementation and application to a variety of practical problems involving either affine-linear or rational dependencies see [12] and the literature cited therein. A more efficient enclosure method is proposed in [7] for the special case of rank-one uncertainty structures. The concepts of self-validating methods and their mathematical background are discussed in many works, see [17] or [15], [16]. Mathematical rigor in the computer arithmetic using directed rounding, in algorithm design, and in program execution allow to guarantee that the hypotheses of suitable inclusion theorems are (or are not) satisfied and thus to guarantee that the stated problem has (or does not have) a solution in an enclosing interval. Note that self-validating methods for linear systems prove existence and uniqueness of the solution for $p \in [p]$ and therefore the non-singularity of $A(p)$.

2.1. Global monotonicity proof

Taking the partial derivative $\frac{\partial}{\partial p_\nu}$ on both sides of (1) we obtain

$$\frac{\partial A(p)}{\partial p_\nu} x(p) + A(p) \frac{\partial x(p)}{\partial p_\nu} = \frac{\partial b(p)}{\partial p_\nu}, \tag{5}$$

where the partial derivatives are applied to vectors and matrices componentwise. Denote $\Delta^\nu := \frac{\partial x(p)}{\partial p_\nu}$, $b^\nu(p) := \frac{\partial b(p)}{\partial p_\nu}$, $A^\nu(p) := \frac{\partial A(p)}{\partial p_\nu}$, where the superscripts do not mean power but corresponding ν-th derivative vector or matrix. Assuming that $[x^*] \in \mathbb{IR}^n$ is a solution enclosure generated by a self-verified parametric solver, $[x^*] \supseteq \square\Sigma^p$, from (5) we obtain the interval linear system

$$A(p)\Delta^\nu = b^\nu(p) - A^\nu(p)[x^*], \tag{6}$$

involving the original parametric matrix and a right-hand-side vector depending on the original parameters and the interval vector of the initial solution enclosures. If $\mathcal{X}^\nu$ denotes the solution set of (6), enclosing it by any self-validated numerical method will give us

$$[\Delta^\nu] \supseteq \square\mathcal{X}^\nu \supseteq \left\{ \frac{\partial x(p)}{\partial p_\nu} \mid p \in [p] \right\}. \tag{7}$$

Thus, if for some $i = 1, \ldots, n$, $0 \notin [\Delta^\nu]_i$ then the corresponding solution component $x_i(p)$ is monotonic with respect to p_ν with a monotonicity type $\text{sign}\left(\frac{\partial x_i(p)}{\partial p_\nu}\right) = \text{sgn}([\Delta^\nu]_i)$. The success of this computer-aided proof will depend on the sharpness of the enclosure (7). To provide a best possible enclosure, the system (6) should be solved by a parametric method. Furthermore, if $A^\nu(p)x$ involves some $x_i, (1 \le i \le n)$ in more than one of the vector components, introducing additional parameters $x_i \in [x^*]_i, (1 \le i \le n)$ will reduce the solution overestimation due to the dependency problem. In order that no exceed parameters are introduced, we define an index set $J^\nu := \{j \mid \{A^\nu(p)\}_{\cdot j} \neq 0\}$, where $\{A^\nu(p)\}_{\cdot j} \neq 0$ means that the jth column of $A^\nu(p)$ is not equal to the zero vector. Denoting $r^\nu(p, x_{J^\nu}) := \frac{\partial b(p)}{\partial p_\nu} - A^\nu(p)x$ and solving the partial derivative system

$$\begin{aligned} A(p) \cdot \Delta^\nu(p, x_{J^\nu}) &= r^\nu(p, x_{J^\nu}) \\ p \in [p], \quad x_{J^\nu} &\in [x^*_{J^\nu}] \end{aligned} \tag{8}$$

by a self-validating parametric solver, will provide a best possible enclosure $[\Delta^\nu]$ for the set of ν-th partial derivatives. Since the quality of the enclosure $[\Delta^\nu]$ in (7) depends also on the sharpness of the initial enclosure $[x^*]$, it might be helpful to resolve the derivative parametric system with an improved enclosure $[x^*]$ of (1). This will be exploited in Section 2.2 and demonstrated in Section 3.

Let for fixed i, $1 \le i \le n$, there exist index sets

$$\begin{aligned} L_+ &:= \{\nu \mid \text{sgn}(\left[\frac{\partial x_i}{\partial p_\nu}\right]) = 1\}, \\ L_- &:= \{\nu \mid \text{sgn}(\left[\frac{\partial x_i}{\partial p_\nu}\right]) = -1\}. \end{aligned}$$

If $L_- \cup L_+ = \{1, \ldots, k\}$ then the exact bounds of $\{\Sigma^p\}_i$ can be obtained by solving two point linear systems, given below, in exact arithmetic or a very sharp floating-point enclosure can be delivered by a self-validated solver for the same two point systems

$$\begin{aligned} [\inf \Sigma^p, \sup \Sigma^p]_i = [\ &\{A^{-1}(p^-_{L_+}, p^+_{L_-}) b(p^-_{L_+}, p^+_{L_-})\}_i, \\ &\{A^{-1}(p^+_{L_+}, p^-_{L_-}) b(p^+_{L_+}, p^-_{L_-})\}_i]. \end{aligned}$$

However, for some i, $1 \le i \le n$, $x_i(p)$ may be not monotonic for p_ν in the domain $[p]$ or the enclosure $[\Delta^\nu]$ is so rough that $0 \in [\Delta^\nu]$, or just $x_i(p)$ may be not dependent of p_ν. This means that there exist index sets L_+, L_-, defined as above, and $L_0 := \{\nu \mid 0 \in [\Delta^\nu]_i\}$ so that

$$L_+ \cup L_- \cup L_0 = \{1, \ldots, k\}, \quad L_0 \neq \emptyset. \tag{9}$$

If (9) holds, considering two new parametric linear systems, involving a reduced number of parameters $p_{L_0} \in [p_{L_0}]$,

$$A^-(p_{L_0})y = b^-(p_{L_0}), \qquad A^+(p_{L_0})z = b^+(p_{L_0}), \tag{10}$$

wherein $a^-_{ij}(p_{L_0}) := a_{ij}(p^-_{L_+}, p^+_{L_-}, p_{L_0})$, $b^-_i(p_{L_0}) := b_i(p^-_{L_+}, p^+_{L_-}, p_{L_0})$, $a^+_{ij}(p_{L_0}) := a_{ij}(p^+_{L_+}, p^-_{L_-}, p_{L_0})$,

$b_i^+(p_{L_0}) := b_i(p_{L_+}^+, p_{L_-}^-, p_{L_0})$, a sharper enclosure of $\{\Sigma^p\}_i$ can be obtained as

$$[\inf \Sigma^p,\ \sup \Sigma^p]_i \subseteq [y^*]_i \cup [z^*]_i \subseteq [x^*]_i.$$

Next we present how in case of (9) it is still possible to proof monotonicity of $x_i(p)$ with respect to p_{L_0} or to proof combinatorial bounds for $\{\Sigma^p\}_i$.

2.2. Local monotonicity proof

Let $[y^*], [z^*]$ be enclosures of the solution sets of the corresponding parametric systems (10). These enclosures usually give an overestimated enclosure of $\Sigma(A(p), b(p), [p])$ but they can be used in an attempt to prove monotonicity properties of $x(p)$ in the local domains $(p_{L_+}^-, p_{L_-}^+, [p_{L_0}])$, $(p_{L_+}^+, p_{L_-}^-, [p_{L_0}])$, respectively. To this end, let fix $q \in L_0$. Taking the partial derivative $\frac{\partial}{\partial p_q}$ on the equations (10), we solve the following derivative systems corresponding to (10) and the parameter $p_q \in p_{L_0}$

$$\begin{aligned} A^-(p_{L_0})\Delta^{y,q} &= r^{y,q}(p_{L_0}, y_{J^{y,q}}) \\ A^+(p_{L_0})\Delta^{z,q} &= r^{z,q}(p_{L_0}, z_{J^{z,q}}), \end{aligned}$$

where $\Delta^{y,q} := \frac{\partial y(p_{L_0})}{\partial p_q}$, $r^{y,q}(p_{L_0}, y_{J^{y,q}}) := \frac{\partial b^-(p_{L_0})}{\partial p_q} - A^-(p_{L_0}) y_{J^{y,q}}$, $J^{y,q} := \{j \mid \{\frac{A^-(p_{L_0})}{\partial p_q}\}_{\cdot j} \neq 0\}$; $\Delta^{z,q}$ and $r^{z,q}(p_{L_0}, y_{J^{z,q}})$ are defined analogously with respect to z. Now, if $[\Delta^{y,q}]$, $[\Delta^{z,q}]$ are the corresponding solution enclosures and $0 \notin [\Delta^{y,q}]_i$, $0 \notin [\Delta^{z,q}]_i$, define $\lambda := \mathrm{sgn}([\Delta^{y,q}]_i)$, $\mu := \mathrm{sgn}([\Delta^{z,q}]_i)$. Thus, $\{\inf \Sigma(A(p), b(p), [p])\}_i$ will be determined by the endpoint p_q^λ of $[p_q] := [p_q^-, p_q^+]$ and $\{\sup \Sigma(A(p), b(p), [p])\}_i$ will be determined by p_q^μ. Note that $\lambda = \mu$ means that $x_i(p)$ is monotone w.r.t. p_q while $\lambda \neq \mu$ means that the corresponding boundaries are combinatorial. In the latter case it may happen that $0 \notin [\Delta^{y,q}]_i$ but $0 \in [\Delta^{z,q}]_i$ and vice-versa.

To be more clear, below we give an algorithm for the local monotonicity procedure, where ParametricSSolve is a function providing guaranteed enclosure of a parametric linear system and globalMonotoneType is a function performing global monotonicity proof and delivering an $n \times k$ matrix monType with elements -1, 1, or 0.

localMonotoneType($Ap, bp, [p]$, monType)
For $i = 1, \ldots, n$
  If $0 \notin \{\text{monType}\}_i$ then $\{\text{mT}\}_i = \{\text{monType}\}_i$
  else begin
    decompose $\{\text{monType}\}_i$ into L_+, L_-, L_0
    generate $A = (a_{ij})$ by $a_{ij} := \{Ap\}_{ij}(p_{L_+}^-, p_{L_-}^+, p_{L_0})$
      $b = (b_i)$ by $b_i := \{bp\}_i(p_{L_+}^-, p_{L_-}^+, p_{L_0})$
    $[s^*] = \text{ParametricSSolve}(A, b, [p_{L_0}])$
    $y = \{\text{globalMonotoneType}(A, b, [p_{L_0}], [\text{s}^*])\}_i$
    generate $A = (a_{ij})$ by $a_{ij} := \{Ap\}_{ij}(p_{L_+}^+, p_{L_-}^-, p_{L_0})$
      $b = (b_i)$ by $b_i := \{bp\}_i(p_{L_+}^+, p_{L_-}^-, p_{L_0})$
    $[s^*] = \text{ParametricSSolve}(A, b, [p_{L_0}])$
    $z = \{\text{globalMonotoneType}(A, b, [p_{L_0}], [\text{s}^*])\}_i$
    For $\nu = 1, \ldots, k$
      $t =$ position of ν in L_0

$$\{\text{mT}\}_{i\nu} := \begin{cases} \{y_t, z_t\} & \text{if } \nu \in L_0 \\ \{\text{monType}\}_{i\nu} & \text{otherwise} \end{cases}$$

    end
  end
end
Return mT

For the sake of an efficient implementation of the proposed approach it should be noted that the original parametric system and the parametric systems of the partial derivatives for all parameters have same matrix and differ only in their right hand side vectors. This allows inverting of only one point matrix and using the same iteration matrix when finding an initial solution enclosure and enclosing the solutions to all partial derivative systems.

2.3. Convexity/concavity proof

Following the considerations along the above lines, one can proof convexity or concavity by the same technique enclosing the solutions of the parametric systems for the second partial derivatives of the original system.

3. Illustrative examples

This section provides some examples illustrating the application of the approach proposed in Section 2. Furthermore, presented are examples demonstrating the advantages and an expanded scope of applications of the approach based on self-validated parametric solvers. In comparison to the approaches based on extreme point computations [1], [2], [8], dealing only with rank-one uncertainty structures, the proposed approach can handle dependencies between $A(p)$ and $b(p)$ as well as nonlinear dependencies. Combinatorial bounds of the parametric solution set can be proven as well as the respective properties of non-square linear parametric problems. The computational efficiency is also discussed.

First we present two problems from structural mechanics which are chosen to be small so that the structure of the parameter dependencies in the corresponding systems is visible. In [13] the presented approach is applied to larger systems coming from finite element approximation of mechanical problems. Namely, combinatorial property of the parametric solution set hull is proven for parametric systems with nonlinear dependencies modeling a one-bay steel

frame and a two-bay two-story frame, the latter involving 18 equations and 13 uncertain parameters. The third example, considered in Section 3.3, involves 81 equations and 101 uncertain parameters. It should be noted that when the concerned parametric linear system is a result of discretized equation of a continued problem, e.g. the finite element discretization of a partial differential equation, proving a certain property of the linear system would not mean the same property for the original infinite dimensional problem. It is beyond the scope of this method to account for the discretization error of the mathematical model in addition to the uncertainty in the parameters, although there are some recent investigations in this direction.

The numerical computations are performed in the environment of *Mathematica* [18] where numerous functions are developed particularly for solving linear parametric problems with uncertain data. The initial version of a *Mathematica* package `IntervalComputations`LinearSystems` [9] supported functions for verified solving of non-parametric interval linear systems and parametric linear systems involving affine-linear dependencies. Latter on the package was extended by a function for verified solving of linear systems whose input data are rational functions of interval parameters [12], [13] and functions allowing efficient handling of non-square (over- or underdetermined) linear parametric systems [11]. All self-verified solvers optionally deliver an inner estimation of the computed outer enclosure of the solution set hull so that the quality of the latter can be estimated. Expanding further the above functionality, several functions, supporting the approach presented in Section 2, are implemented: For a given parametric system (1–3) and an enclosure `xp` of the parametric solution set, `globalMonotoneType(Ap,bp,pars,xp)` is a function that solves k parametric systems (8) and delivers an $n \times k$ matrix with elements $-1, 1$, or 0 denoting the corresponding partial derivative signs, 0 means that monotonicity cannot be proven. For a given parametric system (1–3) and an $n \times k$ matrix `monType`, containing information about the global monotonicity properties of the solution and involving zero elements, `localMonotoneType(Ap,bp,pars,monType)` verifies local monotonicity properties of the parametric solution and delivers an $n \times k$ table whose elements can be $0, 1, -1$, or $\{\lambda, \mu\}$, where $\lambda, \mu \in \{0, 1, -1\}$, corresponding to global monotonicity, or combinatorial, or both parametric solution properties. For a given parametric system (1–3) and an $n \times k$ table `monType`, containing information about the monotonicity and/or combinatorial properties of the solution, `monotoneParametricSolve(Ap,bp,pars,monType)` is a function which based on the information in `monType` delivers a guaranteed enclosure of the parametric solution set. The elements of `monType` can be $0, 1, -1$, or $\{\lambda, \mu\}$ where $\lambda, \mu \in \{0, 1, -1\}$.

The implementation of all parametric solvers is based on a general purpose self-validated parametric method proposed by S. Rump [16] and generalized in [10]. Although the computations presented below in this section use these solvers (the function `ParametricSSolve`), the proposed approach can be applied by using other self-validated parametric solvers, e.g. one based on method proposed in [7].

3.1. Planar frame

Consider a simple planar frame with three types of support and an external load distributed uniformly along the beam as shown in Figure 1. The frame is modelled by using

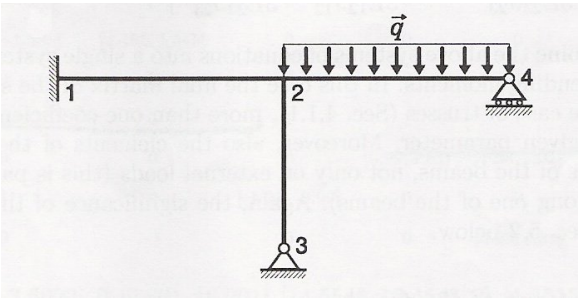


Figure 1. Planar frame after [4].

the method of forces and assuming small displacements and linear elastic material law (for more details see [4], [12]). It is assumed that all beams have the same Young modulus E but momentum of inertia J of the beam cross-sections are related by the formula $J_{12} = J_{23} = 1.5J_{24}$. The lengths of the beams and the load are considered to be uncertain with $l_{12}, l_{24} \in [0.95, 1.05]$, $l_{23} \in [0.7125, 0.7875]$, $q \in [7.5, 12.5]$.

Bounds for the moments and the reactions in the planar frame model should be obtained by solving the following linear system

$$\begin{pmatrix} 2l_{12} & l_{12} & 0 & 0 & 0 \\ l_{12} & 2l_{12}+2l_{23} & -2l_{23} & 0 & 0 \\ 0 & -2l_{23} & 2l_{23}+3l_{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 & l_{12} \\ -1 & 1 & 0 & -l_{12} & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ l_{12}+l_{24} & 0 & l_{23} \\ 0 & 0 & 0 \\ l_{24} & 0 & 0 \end{pmatrix} x = \begin{pmatrix} 0 \\ 0 \\ -\frac{3}{8}ql_{24}^3 \\ 0 \\ ql_{24} \\ ql_{24}l_{12}+\frac{1}{2}l_{24} \\ 0 \\ \frac{1}{2}ql_{24}^2 \end{pmatrix}.$$

Note that the right-hand side depends on the beam lengths, not only on the external load (this is partly due to the presence of distributed load along one of the beams), and the dependence is nonlinear. Our goal is to prove monotonicity properties of the system solution or some combinatorial properties of the solution which will allow a sharp enclosure of the system response. We assume that the required *Mathematica* functions are loaded in the memory and that the variables `mat, vec, pars` contain the parametric matrix, the r.h. side vector and the 4-tuple of parameters $(l_{12}, l_{23}, l_{24}, q)$ with their interval values, respectively. Note that the computer algebra environment of *Mathematica* allows the same mathematical (symbolic) notations, as in the above definition of the system, to be used for entering the input data. Our approach requires that first we find an initial enclosure of the parametric solution set and then execute the function `globalMonotoneType`.

```
In[2]:=xp=ParametricSSolve[mat, vec, pars];
    mT1 = globalMonotoneType[mat,vec,pars,xp]
Out[3]={{0, 1,0, 1},{1,-1,0,-1},
        {1,-1,0,-1},{1,-1,0,-1},{-1,1,0,1},
        {1, 0,0, 1},{0, 0,0,-1}, {0,0,0,1}}
```

Thus, for each solution component we managed to prove monotonic dependence with respect to some of the parameters. The information about that dependence (`mT1`) is used by the function `localMonotoneType` to compute sharper solution enclosures for the initial parametric system and to prove local monotonicity properties of the derivative systems corresponding to the lower and upper solution set bounds for each of the solution components.

```
In[4]:= mT2=localMonotoneType[mat, vec,
                pars, mT1] /.  {{t_, t_}->t}
Out[4]={{-1,1,1,1}, {1,-1,-1,-1},
        {1,-1,1,-1},{1,-1,-1,-1},{-1,1,1,1},
        {1,1,1,1}, {0,-1,1,-1}, {0,1,-1,1}}
```

The proven local monotonicity is of the same type for both derivative systems which means corresponding global monotonicity properties. Unfortunately, the dependence of the last two solution components w.r.t. the first parameter is still not known. However, we can use again the improved knowledge (`mT2`) about the solution global monotonicity properties.

```
In[5]:= localMonotoneType[mat, vec, pars,
                      mT2] /.  {{t_, t_}->t}
Out[5]={{-1,1,1,1}, {1,-1,-1,-1},
       {1,-1,1,-1},{ 1,-1,-1,-1},{-1,1,1,1},
       {1, 1,1, 1},{-1,-1, 1,-1},{1,1,-1,1}}
```

Thus global monotonicity properties of the parametric solution are rigorously proven which allows finding the exact bounds or a guaranteed very sharp enclosure for the parametric system response.

3.2. Cantilevar beam

Consider the cantilevar beam shown in Figure 2. Each

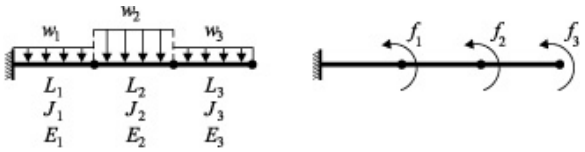


Figure 2. Cantilevar beam.

beam element is characterized by its length L_i, modulus of elasticity E_i, and momentum of inertia J_i, and is loaded with uniformly distributed load w_i, $i = 1, 2, 3$. It is assumed that the axial deformations are neglected. All the parameters are considered to be uncertain. The beam is modelled by the following linear system

$$\begin{pmatrix} \frac{4p_1}{L_1}+\frac{4p_2}{L_2} & \frac{2p_2}{L_2} & 0 \\ \frac{2p_2}{L_2} & \frac{4p_2}{L_2}+\frac{4p_3}{L_3} & \frac{2p_3}{L_3} \\ 0 & \frac{2p_3}{L_3} & \frac{4p_3}{L_3} \end{pmatrix} x = \begin{pmatrix} -\frac{w_1L_1^2}{12}+\frac{w_2L_2^2}{12} \\ -\frac{w_2L_2^2}{12}+\frac{w_3L_3^2}{12} \\ -\frac{w_3L_3^2}{12} \end{pmatrix},$$

where $p_1, p_2, p_3 \in [0.018905, 0.021105]$, $p_i = E_iJ_i$, $i = 1, 2, 3$, represent variations in the material properties, the element lengths are $L_1 \in [9.95, 10.05]$, $L_2 \in [7.96, 8.04]$, $L_3 \in [5.97, 6.03]$, and the distributed loads w_1, w_2, w_3 vary independently within $[0, 4]$. The goal is to prove solution properties allowing a sharp enclosure of the system response. We apply our approach as in the previous example but with new input data. The 9-tuple of parameters is $(p_1, p_2, p_3, L_1, L_2, L_3, w_1, w_2, w_3)$.

```
In[7]:=xp=ParametricSSolve[mat, vec, pars];
     mT1=globalMonotoneType[mat,vec,pars,xp]
Out[8]= {{0,0,0,0,0,0,-1, 1,-1},
         {0,0,0,0,0,0, 1,-1, 1},
         {0,0,0,0,0,0,-1, 1,-1}}
```

It is proven a monotonic dependence of the parametric solution with respect to the last three (load) parameters w_1, w_2, w_3. Due to this we can check the local monotonicity of the corresponding derivative system couples and we get the following result

```
In[9]:= localMonotoneType[mat,vec,pars,mT1]
Out[9]={
{{1,-1},{1,-1},{1,-1},{-1,1},{-1,1},{-1,1},-1,1,-1},
{{1,-1},{1,-1},{1,-1},{-1,1},{-1,1},{-1,1},1,-1,1},
{{1,-1},{1,-1},{1,-1},{-1,1},{-1,1},{-1,1},-1,1,-1}}
```

which shows the combinatorial property of the parametric solution set. Furthermore, to compute the solution set hull there is no need to solve $2^9 = 256$ point linear systems corresponding to all possible combinations of the interval end-points but only those 4 corresponding to the global and local monotonicity types, obtained in `Out[9]`.

3.3. Truss Cantilevar

As a large example we consider the truss cantilevar shown in Figure 3 and proposed as a benchmark example by R. Muhanna [5]. A finite element model for a one-bay

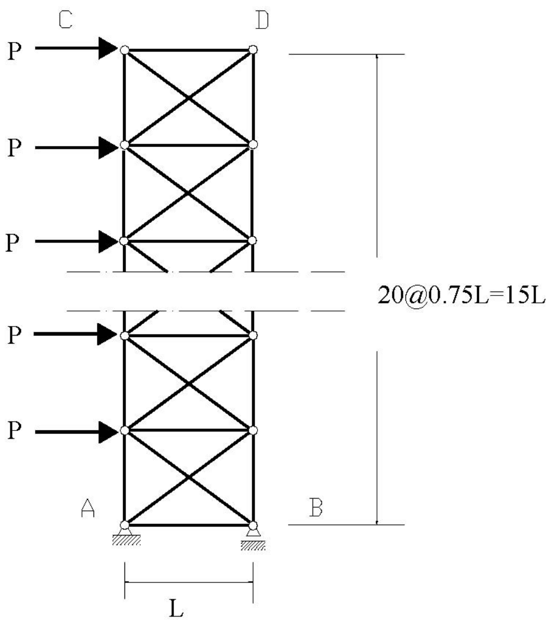


Figure 3. Truss cantilevar after [5].

20-floor truss cantilever consists of 42 nodes and 101 elements. The modulus of elasticity E_i of each element is considered to be uncertain. A complete description of the problem can be found in [5]. The model leads to a parametric linear system involving 81 variables and 101 uncertain parameters E_i connected by affine-linear dependencies. The computer-assisted approach based on a self-verified parametric solver is applied for proving the combinatorial property of the solution set hull for 2% parameter uncertainties, that is $E_i \in [0.99, 1.01]$, $i = 1, \ldots, 101$.

The computer-aided proof was slow for such a problem involving large number of uncertain parameters. Some experiments showed that ignoring the rounding errors in the calculations not explicitly involving intervals (approach used by many authors for the sake of speed) while solving the parametric systems leads to some wrong monotonicity types. This example demonstrates the necessity of a fully rigorous implementation of the parametric solvers used for a computer-aided proof. Monotonicity or combinatorial properties of most solution components are proven. Here we present the properties of the last two solution components corresponding to the displacements in horizontal and vertical direction of the upper right corner (D, in Figure 3), which is the problem of interest. While the solution component 80 is monotone with respect to all parameters: (-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1), the last solution component possesses the combinatorial property w.r.t. E_{11}, that is (1, 1, 1, 1, -1, -1, 1, 1, 1, -1, {-1, 1}, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1).

Although it can be theoretically proven for some problems that the extremal values of any solution component are attained at vertices of the parameter box, which are these vertices is not known. Thus, finding the exact solution set hull by combinatorial extreme point calculations may be prohibitive for large number of parameters (truss cantilever system requires solving approx. 2.5353×10^{30} point systems), while the proof based on verified parametric solvers, although heavy and slow, is usually feasible providing in addition explicit information for the monotonicity types.

3.4. Properties of least squares solutions

It was mentioned in [2], as an interesting open problem, the extent to which least squares solutions

$$x^{\mathrm{LS}}(p) = \left(A^\top(p)A(p)\right)^{-1} A^\top(p)b(p),$$

where $A(p)$ is an $n \times m$ matrix with rank $A(p) = m < n$ for all admissible $p \in [p]$, are monotonic, convex or concave with respect to the uncertain model parameters. Results along these lines would be important when studying the dependence of the least squares solution on either design variables or uncertain parameters.

Non-parametric interval least squares estimates are considered by several authors. In [3] a solution enclosure is sought by solving the normal equation $A^\top Ax = A^\top b$. Because the spectral radius of the matrix is greater than 1 for most problems, interval enclosures are obtained by a specially designed methodology based on QR decomposition. The overestimation of the obtained enclosures is assessed by computing the convex hull of the solution basing on the signs of the partial derivatives of the solution with respect to the elements of the matrix and the right hand side vector. The utilized monotonicity approach is almost the same as the exploited in this work with the important difference that the sign of the partial derivatives is evaluated at the interval mid points and not proven over the interval box. It is well known that such point estimates may be not correct in general.

To avoid the ill-conditioning of $A^\top(p)A(p)$ in enclosing the solutions of non-square over- or underdetermined parametric linear systems, we follow the proposal of S. Rump [15] for nonparametric systems, and consider a corresponding augmented square linear system. For the least squares problem, the augmented system is

$$\begin{pmatrix} A(p) & -I \\ 0 & A^\top(p) \end{pmatrix} \cdot \begin{pmatrix} x^{\mathrm{LS}} \\ y \end{pmatrix} = \begin{pmatrix} b(p) \\ 0 \end{pmatrix}. \quad (11)$$

The augmented system, being parametric even for nonparametric nonsquare systems, helps also for reducing the dependencies in the parametric normal equation. How to handle efficiently the parameter dependence in the above augmented system is detailed in [11]. Thus, for a rigorous computer-assisted proof, the corresponding derivative systems should be constructed from the augmented system (11) and their solutions should be enclosed by the algorithms presented in the preceding sections.

3.5. Computational efficiency

Beside the expanded scope of applications of the presented approach, it is computationally cheaper compared to the extreme point computations [2], [8]. For rank-one uncertainty structures, checking monotonicity properties of the parametric solution, as proposed in [2], depends exponentially on the number of the parameters and requires about $\alpha 2^k$ solutions of point linear systems, where α is a factor depending on the implementation. Proving the monotonicity properties by the presented approach, based on self-validated parametric solvers, requires solving of $1+k$ parametric interval linear systems. In case that local monotonicity should be additionally proven, the number of parametric solvers increases by $2(n+1)$ for each execution of this procedure.

Based on the available self-validated parametric solver, the success of a proof depends on the quality of that solver, that is its ability to provide sharp solution enclosures. For example, the general-purpose parametric fixed-point iteration [16] is convergent only for strongly regular parametric matrices and may fail for some $A(p)$ which is regular but not strongly regular on $[p]$. It is also well-known that this interval parametric method produces quite overestimated solution enclosures, and even fail, for very large parameter intervals. On another side, a recently developed method [7] is extremely efficient for enclosing the solution set of large scale parametric systems involving rank-one uncertainty structures and wide parameter intervals.

4. Conclusion

We demonstrated the application of self-verified parametric solvers for computer-assisted numerical proof of some properties of the parametric solution. Namely, monotonicity and convexity/concavity dependence of the solution components with respect to the system parameters, as well as extreme point results for the convex hull of the parametric solution set can be proven by guaranteed floating-point computations. The methodological novelties of the proposed technique reside at two places of the paper: First, the algorithm for proving monotonicity of the solution components requires solving a parametric linear system (with the same matrix) involving additional parameters in the right-hand side, the parameters corresponding to the components of the initial solution enclosure. This provides sharper enclosure of the derivatives, and thus the success of the method (without necessity of interval subdivision). Second, the combinatorial hull of the parametric solution set is achieved by proving local monotonicity properties of the solution components.

The proposed technique, based on self-validated parametric solvers, does not have the limitations of the extreme point methods [2], [8] for rank-one uncertainty structures, allows handling of more general parameter dependencies and non-square parametric linear systems at a cheaper price providing in addition guaranteed results. The numerical proof of parametric solution properties is applicable to problems formulated in terms of different uncertainty theories which rely on interval arithmetic for computations, such as deterministic interval uncertainties, fuzzy set theory, random set theory, probability bounds theory, or mixed type uncertainties.

Acknowledgements. This work was partially supported by the Bulgarian National Science Fund under grant No. MM1301/03.

References

[1] A. Dreyer. *Interval Analysis of Analog Circuits with Component Tolerances*. PhD thesis, Kaiserslautern Univ. of Technology, Kaiserslautern, 2005.

[2] A. Ganesan, S. Ross, B. Ross Barmish. An Extreme Point Result for Convexity, Concavity and Monotonicity of Parametrized Linear Equation Solutions. *LAA*, 390:61–73, 2004.

[3] D. M. Gay. Interval Least Squares — a Diagnostic Tool. In R. E. Moore (Ed). Reliability in Computing, *Perspectives in Computing* 19, Academic Press, San Diego, 1988, 183–205.

[4] Z. Kulpa, A. Pownuk, I. Skalna. Analysis of Linear Mechanical Structures with Uncertainties by Means of Interval Methods. *CAMES* 5:443–477, 1998.

[5] R. L. Muhanna. Benchmarks for Interval Finite Element Computations. Web site, 2004. http://www.gtsav.gatech.edu/rec/ifem/benchmarks.html

[6] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[7] A. Neumaier, A. Pownuk. Linear Systems with Large Uncertainties, with Applications to Truss Structures, November 2004. http://www.mat.univie.ac.at/ neum/ms/linunc.pdf

[8] E. D. Popova. Quality of the Solution Sets of Parameter-Dependent Interval Linear Systems. *ZAMM*, 82(10):723–727, 2002.

[9] E. D. Popova. Parametric Interval Linear Solver. *Numerical Algorithms*, 37(1–4):345–356, 2004.

[10] E. D. Popova. Generalizing the Parametric Fixed-Point Iteration. *Proceedings in Applied Mathematics & Mechanics* (PAMM) 4(1):680–681, 2004.

[11] E. D. Popova. Improved Solution Enclosures for Over- and Underdetermined Interval Linear Systems. In I. Lirkov, S. Margenov, J. Wasniewski (Eds). Proceedings of LSSC 2005, *Lecture Notes in Computer Science* 3743, 2006, 305–312.

[12] E. D. Popova. Solving Linear Systems whose Input Data are Rational Functions of Interval Parameters. Preprint 3/2005, Institute of Mathematics and Informatics, BAS, Sofia, 2005. (http://www.math.bas.bg/~epopova/papers/05PreprintEP.pdf)

[13] E. D. Popova, R. Iankov, Z. Bonev. Bounding the Response of Mechanical Structures with Uncertainties in All the Parameters. In: R. Muhannah, R. Mullen (Eds). Proc. of the NSF Workshop on Reliable Engineering Computing, Svannah, Georgia, USA, Feb. 22-24, 2006, 245–265.

[14] J. Rohn. Linear Interval Equations with Dependent Coefficients. Symposium "Interval Methods for Numerical Computations", Oberwolfach, 1990. In: J. Rohn. A Method for Handling Dependent Data in Interval Linear Systems. Technical Report No. 911, Institute of Computer Science, Academy of Science of the Czech Republic, July 2004.

[15] S. Rump. Solving Algebraic Problems with High Accuracy. In: U. Kulisch and W. Miranker (Eds). *A New Approach in Scientific Computation*. Academic Press, 1983, 51–120.

[16] S. Rump. Verification methods for dense and sparse systems of equations. In: J. Herzberger (Ed). *Topics in Validated Computations*. Elsevier Science B. V., 1994, 63–135.

[17] S. Rump. Computer-assisted Proofs and Self-validating Methods. In: Bo Einarsson (Ed). *Accuracy and Reliability in Scientific Computing*. SIAM, 2005, 195–240.

[18] S. Wolfram. *The Mathematica Book*. Wolfram Media/Cambridge U. Press, 4th ed., 1999.

VALENCIA-IVP: A Comparison with Other Initial Value Problem Solvers

Andreas Rauh, Eberhard P. Hofer
Institute of Measurement, Control,
and Microtechnology
University of Ulm
D-89069 Ulm, Germany
{Andreas.Rauh, EP.Hofer}@uni-ulm.de

Ekaterina Auer
Faculty of Engineering, IIIS
University of Duisburg-Essen
D-47048, Germany
auer@inf.uni-due.de

Abstract

Validated integration of ordinary differential equations with uncertain initial conditions and uncertain parameters is important for many practical applications. If guaranteed bounds for the uncertainties are known, interval methods can be applied to obtain validated enclosures of all states. However, validated computations are often affected by overestimation, which, in naive implementations, might even lead to meaningless results. Parallelepiped and QR preconditioning of the state equations, Taylor model arithmetic, as well as simulation techniques employing splitting and merging routines are a few existing approaches for reduction of overestimation. In this paper, the recently developed validated solver VALENCIA-IVP *and several methods implemented there for reduction of overestimation are described. Furthermore, a detailed comparison of this solver with* COSY VI *and* VNODE*, two of the most well-known validated ODE solvers, is presented. Simulation results for simplified system models in mechanical and bioprocess engineering show specific properties, advantages, and limitations of each tool.*

1 Introduction

In this paper, a detailed comparison of different validated solvers for initial value problems (IVPs) for ordinary differential equations (ODEs) is presented. The performance of the interval arithmetic solver VALENCIA-IVP (***VAL***idation of state ***ENC***losures using ***I***nterval ***A***rithmetic for ***I***nitial ***V***alue ***P***roblems[1]) proposed by the authors in [1], is compared with COSY VI [9] and VNODE [10]. For this comparison, the dynamics of a double pendulum and a subsystem model of biological wastewater treatment processes are analyzed as representative examples. Since validated ODE solvers allow for computation of enclosures which are guaranteed to contain all possible system states, they are advantageous compared to stochastic or grid-based techniques if the behavior of safety critical systems or the worst-case influence of parameter uncertainties are of interest.

The basic principle of VALENCIA-IVP is to describe state enclosures by non-validated approximate solutions of the IVP and the corresponding guaranteed error bounds. These bounds are determined by a fixed-point iteration without separate calculation of discretization errors. The main difference to commonly used validated methods for simulation of dynamical systems [7, 9, 10] is that only the first partial derivatives of the ODEs with respect to states, parameters, and time are required. These first derivatives are necessary for mean-value rule evaluation as well as advanced interval techniques such as monotonicity tests, splitting, and merging of subintervals which are applied to reduce overestimation in VALENCIA-IVP. First results in the context of multibody modeling and simulation software [1] have shown that the achievable simulation quality of VALENCIA-IVP is comparable to results of VNODE.

A new extension of VALENCIA-IVP is presented in this paper which excludes state intervals resulting from overestimation by a consistency test based on backward integration. Additionally, this exclusion strategy is combined with existing splitting and merging techniques. The selected ODE solvers are compared with respect to their specific algorithmic properties, the achievable simulation qualities, and the required CPU times for exemplarily chosen applications.

In Section 2, the problem of validated simulation of continuous-time systems with uncertain initial states and uncertain parameters is formulated. Additionally, the main algorithmic properties of VNODE and COSY VI are summarized briefly. In Section 3, the basic algorithm of VALENCIA-IVP is described and a proof of the conservativeness of the obtained interval bounds is presented. The focus of Section 4 is a consistency test for VALENCIA-IVP aiming at the reduction of overestimation. In Section 5, sim-

[1] http://www.valencia-ivp.com

ulation results for the three above-mentioned ODE solvers are compared using two examples. It is shown that the basic version of VALENCIA-IVP could reduce CPU time in comparison to COSY VI and overestimation in comparison to VNODE for several important scenarios. The paper is concluded in Section 6 by an outlook on future research.

2 Validated Solvers for Ordinary Differential Equations

2.1 Problem Formulation

The goal of this article is to compare validated solvers for IVPs described by time-varying ODEs

$$\dot{x}_s(t) = f_s(x_s(t), p(t), t) \tag{1}$$

with the state vector $x_s(t) \in \mathbb{R}^{n_s}$, the vector $p(t) \in \mathbb{R}^{n_p}$ of (uncertain) system parameters, and the nonlinear state-space representation $f_s : D \mapsto \mathbb{R}^{n_s}$, $D \subset \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \times \mathbb{R}^1$. The initial states are denoted by $x_s(0) = x_s^0$. Since in most practical applications only conservative bounds of the initial conditions and the system parameters are known, they are assumed to be bounded by the intervals

$$[x_s^0] := \left[\underline{x}_s^0 \; ; \; \overline{x}_s^0\right] \text{ and } [p(t)] := \left[\underline{p}(t) \; ; \; \overline{p}(t)\right] \quad . \tag{2}$$

Furthermore, the system parameters p may be time-varying which is expressed by the additional differential equation

$$\dot{p}(t) = \Delta p(t) \quad , \tag{3}$$

where the variation rates

$$\Delta p(t) \in [\Delta p(t)] := \left[\underline{\Delta p}(t) \; ; \; \overline{\Delta p}(t)\right] \tag{4}$$

do not have to be known exactly. To simplify the application of validated ODE solvers, an extended state vector $x(t) = \left[x_s^T(t) \quad p^T(t)\right]^T \in \mathbb{R}^n$, $n = n_s + n_p$, consisting of the original system states and parameters is introduced. The resulting state-space representation is then denoted by

$$\dot{x}(t) = f(x(t), t) = \begin{bmatrix} f_s(x_s(t), p(t), t) \\ \Delta p(t) \end{bmatrix} \tag{5}$$

with $f : D \mapsto \mathbb{R}^n$, $D \subset \mathbb{R}^n \times \mathbb{R}^1$. Treating the time variable t separately from the extended state vector $x(t)$ is especially advantageous for modeling of dynamics of time-varying systems for which fixed bounds of states and/or parameters, e.g. as defined in (2), are known.

2.2 VNODE

To use VNODE, it is necessary to discretize the time horizon and transform the given IVP into an autonomous one. The state equations f are assumed to be continuously differentiable up to a given order $(q-1) > 0$. The algorithm of VNODE consists of two stages [10]:

Stage One. Existence and uniqueness of the solution of an IVP is proven by calculation of guaranteed a priori enclosures of all reachable states in the time interval between two subsequent discretization points. This is done with the help of Banach's fixed point theorem and the high order enclosure method [11] which generalizes the usual techniques based on the Picard iteration [8].

Stage Two. A tight enclosure of the solution is computed using the interval enclosure at the preceding time step and the local error which encloses all discretization errors at the current step. Here, VNODE offers the choice between the direct Taylor series algorithm, Lohner's QR-factorization algorithm, and the interval Hermite-Obreschkoff algorithm. All methods make use of the mean value theorem to compute tighter ranges of Taylor coefficients. Owing to the open structure of this object oriented solver, new algorithms can be easily added to its core.

VNODE uses PROFIL/BIAS [5] for interval arithmetic and FADBAD/TADIFF [2,3] to obtain Taylor coefficients and their Jacobians. Growth of the computed interval diameters with progressing simulation time is inevitable as long as only explicit integration techniques are applied.

2.3 COSY VI

The Taylor model based ODE solver COSY VI performs high order Taylor expansions of the solution in time and initial conditions to reduce overestimation [9]. It seeks to improve conventional validated solvers with respect to modeling of the local functional behavior and control of the long-term growth of integration errors. The first task is solved using the Picard iteration in combination with Schauder's fixed-point theorem. The long-term growth of integration errors is controlled by the shrink wrapping method, which minimizes the interval remainder of the Taylor model of the solution using appropriate transformations of the polynomial part, and with the help of different preconditioning methods, which consider the Taylor model in appropriate coordinate systems.

If suitable orders and step sizes are chosen, a significant reduction of overestimation is possible for highly nonlinear systems [4]. A drawback of this solver is high computational time for systems with many variables.

3 VALENCIA-IVP

In VALENCIA-IVP, the validated enclosures of all reachable states are assumed to be defined by

$$[x_{encl}(t)] = x_{app}(t) + [R(t)] \quad , \tag{6}$$

where $x_{app}(t)$ denotes an arbitrary non-validated approximate solution to the IVP. Validated bounds $[R(t)]$ of the approximation errors are determined by the iteration scheme presented in the following Subsection.

3.1 Iteration Scheme and Proof for Conservativeness of the State Enclosures

Most validated techniques to enclose solutions of IVPs rely on integration of the set of ODEs $\dot{x}(t) = f(x(t), t)$ on a finite time interval according to

$$\begin{aligned} x(t) &= x(0) + \int_0^t f(x(\tau), \tau)\, d\tau \\ &\subseteq x(0) + [0\ ;\ t] \cdot f([B], [0\ ;\ t]) \end{aligned} \tag{7}$$

with $t \in [0\ ;\ T]$ and $x(0) \in \left[x^0\right]$. The range $f([B], [0\ ;\ t])$, which is obtained after substituting the bounding box $[B]$ enclosing all reachable states in the considered time interval $[0\ ;\ t]$ for $x(\tau)$ in the integrand in (7), represents a conservative enclosure of all possible time derivatives of $x(t)$ in this time interval. This property as well as the Picard iteration

$$\left[B^{(\kappa+1)}\right] = \left[x^0\right] + [0\ ;\ t] \cdot f\left(\left[B^{(\kappa)}\right], [0\ ;\ t]\right) \quad , \tag{8}$$

which is usually applied to calculate *time-invariant* bounding boxes $[B]$, are necessary for derivation and proof of the iteration scheme of VALENCIA-IVP [1]. Superscript indices (κ) denote the number of the iteration step. After initialization of the bounding box with $\left[B^{(0)}\right] = \left[x^0\right]$, the iteration formula is evaluated until $\left[B^{(\kappa+1)}\right] \approx \left[B^{(\kappa)}\right]$ if $\left[B^{(1)}\right] \subseteq \left[B^{(0)}\right]$. Otherwise, the initial interval box $\left[B^{(0)}\right]$ has to be inflated until this inclusion property holds. Almost all information about the system dynamics is disregarded if the interval bounds $[B]$ are assumed to be *constant*. Hence, the replacement of $[B]$ by time-varying state enclosures (6) with unknown error terms $[R(t)]$ in VALENCIA-IVP leads to improved state enclosures.

Theorem 1 *Consider an IVP as defined in Subsection 2.1 with $f : D \mapsto \mathbb{R}^n$, $D \subset \mathbb{R}^n \times \mathbb{R}^1$ open, $f \in C^1(D, \mathbb{R}^n)$. Then, all reachable states at the point of time t are contained in the interval enclosure (6), if the error bounds $[R(t)]$ are computed by the following two-stage procedure.*

1. Iterative computation of an interval enclosure of all possible time derivatives $\left[\dot{R}(t)\right]$ of the error term by

$$\begin{aligned} \left[\dot{R}^{(\kappa+1)}(t)\right] &= -\dot{x}_{app}(t) + f\left(\left[x_{encl}^{(\kappa)}(t)\right], t\right) \\ &= -\dot{x}_{app}(t) + f\left(x_{app}(t) + \left[R^{(\kappa)}(t)\right], t\right) \\ &=: r\left(\left[R^{(\kappa)}(t)\right], t\right) \quad . \end{aligned} \tag{9}$$

This iteration converges to a verified enclosure of $\left[\dot{R}(t)\right]$ if $\left[\dot{R}^{(\kappa+1)}(t)\right] \subseteq \left[\dot{R}^{(\kappa)}(t)\right]$. The iteration (9) is continued until $\left[\dot{R}^{(\kappa+1)}(t)\right] \approx \left[\dot{R}^{(\kappa)}(t)\right]$.

2. Verified integration of $\left[\dot{R}^{(\kappa+1)}(t)\right]$, $0 \le t \le T$, with respect to time according to

$$\begin{aligned} \left[R^{(\kappa+1)}(t)\right] &\subseteq \left[R^{(\kappa+1)}(0)\right] + \int_0^t \left[\dot{R}^{(\kappa+1)}(\tau)\right] d\tau \\ &= \left[R^{(\kappa+1)}(0)\right] + \int_0^t r\left(\left[R^{(\kappa)}(\tau)\right], \tau\right) d\tau \end{aligned}$$

replaced by the guaranteed bound

$$\begin{aligned} \left[R^{(\kappa+1)}(t)\right] \subseteq &\left[R^{(\kappa+1)}(0)\right] \\ &+ t \cdot r\left(\left[R^{(\kappa)}([0\ ;\ t])\right], [0\ ;\ t]\right) \quad . \end{aligned} \tag{10}$$

These updated error bounds are required for evaluation of the formula (9) in the next iteration step. Uncertainties of the initial conditions are accounted for by choosing $[R(0)]$ such that $\left[x^0\right] \subseteq x_{app}(0) + [R(0)]$. ■

Proof 1 *Using the Picard iteration (8), a bounding box $[B]$ of all states which are reachable in the time interval $t \in [0\ ;\ T]$ can be determined according to Banach's fixed-point theorem. Substituting $[x_{encl}]$ for the bounding box $[B]$ on both sides of (8) leads to*

$$\begin{aligned} &\left[x_{encl}^{(\kappa+1)}([0\ ;\ T])\right] = \\ &\left[x^0\right] + [0\ ;\ T] \cdot f\left(\left[x_{encl}^{(\kappa)}([0\ ;\ T])\right], [0\ ;\ T]\right) \quad . \end{aligned} \tag{11}$$

Let the approximation error in (6) be defined by

$$[R([0\ ;\ T])] := [R(0)] + [0\ ;\ T] \cdot \left[\dot{R}([0\ ;\ T])\right] \quad , \tag{12}$$

where $\left[\dot{R}([0\ ;\ T])\right]$ is a conservative interval enclosure of all possible time derivatives in the considered time interval. Then, the iteration formula (11) is equivalent to

$$\begin{aligned} &x_{app}([0\ ;\ T]) + \left[R^{(\kappa+1)}([0\ ;\ T])\right] = \\ &\left[x^0\right] + [0\ ;\ T] \cdot f\left(\left[x_{encl}^{(\kappa)}([0\ ;\ T])\right], [0\ ;\ T]\right) \quad . \end{aligned} \tag{13}$$

According to definition (12), $\left[\dot{R}^{(\kappa+1)}([0\ ;\ T])\right]$ is a guaranteed interval enclosure of all possible time derivatives of $\left[R^{(\kappa+1)}([0\ ;\ T])\right]$ in the time interval $[0\ ;\ T]$. Analogously, $f\left(\left[x_{encl}^{(\kappa)}([0\ ;\ T])\right], [0\ ;\ T]\right)$ includes the

time derivative of the right hand side of (13). Therefore, differentiation with respect to time on both sides of (13) and solving for $\left[\dot{R}^{(\kappa+1)}\right]$ *leads directly to the iteration formula (9). Finally, evaluation of the sum of the approximate solution* $x_{app}(t)$ *and the bounds of the approximation error using outward rounding of the resulting interval provides a verified state enclosure of the solution of the IVP.* ■

Since no series expansion of the solution of the IVP is necessary in VALENCIA-IVP, no computation of guaranteed bounds for discretization errors (cf. VNODE) is needed. Note that the interval enclosure $[R]$ could also be obtained without the formulas (9) and (10) by verified integration of the ODE

$$\dot{R}(t) = -\dot{x}_{app}(t) + f\left(x_{app}(t) + R(t), t\right) \tag{14}$$

using an arbitrary validated solver. Here, the approximation errors are obtained by substituting (6) for $x(t)$ in (5).

3.2 Basic Algorithm

In the following, the key components of VALENCIA-IVP are discussed with the focus on computation of suitable approximate solutions $x_{app}(t)$ and methods for reduction of overestimation in the iteration formula (9).

In general, one can use arbitrary non-validated approximations $x_{app}(t)$ in VALENCIA-IVP, which are obtained either analytically or numerically. Without loss of generality, $x_{app}(t)$ is computed for the interval midpoints of the uncertain initial states and system parameters as initial conditions, i.e., $x_{app}(0) = \mathrm{mid}\left(\left[x^0\right]\right) = \frac{1}{2}\left(\underline{x}^0 + \overline{x}^0\right)$.

Analytical approximations can be obtained after linearization of nonlinear ODEs or by neglection of nonlinear terms and can be improved by suitable perturbation techniques. However, especially for high-dimensional problems, **numerical approximations** are often advantageous, since they are more flexible and not restricted to a specific class of state equations.

A non-validated numerical approximation $\left\{x_i^N\right\}$, $i = 0, \ldots, L$, for the original IVP with $x_0^N = \mathrm{mid}\left(\left[x^0\right]\right)$ can be calculated over the grid $\{t_i\}$ with $t_L = T$ by any non-validated IVP solver. Since analytic expressions for $x_{app}(t)$ and its time derivative $\dot{x}_{app}(t)$ are required in (9), they are determined by minimization of a distance measure

$$D = \sum_{i=1}^{L} d\left(x_i^N - x_{app}(t_i)\right) \overset{\text{e.g.}}{=} \sum_{i=1}^{L} \left\|x_i^N - x_{app}(t_i)\right\|_2^2 \ ,$$

for all numerically determined points x_i^N. The current C++ version of VALENCIA-IVP uses the computationally inexpensive linear interpolation

$$x_{app}(t) = x_i^N + \frac{x_{i+1}^N - x_i^N}{t_{i+1} - t_i} \cdot (t - t_i) \tag{15}$$

with the time derivative

$$\dot{x}_{app}(t) = \frac{x_{i+1}^N - x_i^N}{t_{i+1} - t_i} \tag{16}$$

for $t \in [t_i \ ; \ t_{i+1}]$, $i = 0, \ldots, L-1$, where $\left\{x_i^N\right\}$ is computed by an explicit Euler method with constant step size.

The iteration (9) is initialized by choosing interval enclosures for $[R(t)]$ and $\left[\dot{R}(t)\right]$ such that $\left[x^0\right] \subseteq x_{app}(0) + [R(0)]$. The evaluation of (9) is continued if $\left[\dot{R}^{(1)}(t)\right] \subseteq \left[\dot{R}^{(0)}(t)\right]$, see Theorem 1. Otherwise, the interval widths of the initial guesses for $[R(t)]$ and $\left[\dot{R}(t)\right]$ are increased to check if the iteration converges for larger error bounds.

To improve convergence of the iteration and to reduce the width of the error bounds, the time span $[0 \ ; \ T]$ is split into shorter time intervals. Obviously, the validated integration w.r.t. time in formula (10) has to be replaced by

$$\left[R^{(\kappa+1)}(t_{i+1})\right] = \left[R^{(\kappa+1)}(0)\right] + \sum_{j=0}^{i} (t_{j+1} - t_j) \cdot r\left(\left[R^{(\kappa)}([t_j \ ; \ t_{j+1}])\right], [t_j \ ; \ t_{j+1}]\right) \tag{17}$$

for all t_i, $i = 0, \ldots, L-1$. If numerical approximations are used, the grid $\{t_i\}$ (not necessarily equally spaced) is predefined by the non-validated ODE solver.

To obtain tightest possible bounds in evaluation of the iteration formula (9), the intersection of **natural interval evaluation** with **mean-value rule evaluation**

$$r([z]) \in r(z_m) + \left.\frac{\partial r}{\partial z}\right|_{z=[z]} \cdot ([z] - z_m) \quad \text{for all} \quad z \in [z] \ , \tag{18}$$

where

$$[z] = \begin{bmatrix} [R(t_i)] \\ [t_i \ ; \ t_{i+1}] \end{bmatrix} \quad \text{and} \quad z_m = \mathrm{mid}([z]) \ , \tag{19}$$

is computed. Additionally, a **monotonicity test** is performed by VALENCIA-IVP for further reduction of overestimation. If a component r_i, $i = 1, \ldots, n$, is monotonic w.r.t. at least one z_j, $j = 1, \ldots, n+1$, $[z_j]$ can be replaced by its interval bounds during range computation. E.g., if $\inf\left(\frac{\partial r_i}{\partial z_j}\right) > 0$, $[z_j]$ can be replaced by $\underline{z}_j$ to compute the infimum of the range of r_i over $[z]$ and by $\overline{z}_j$ to compute its supremum. All other cases are summarized in Tab. 1.

Optionally, if monotonicity cannot be proven for r_i, all arguments of r_i with interval diameters which are significantly larger than zero can be split into subintervals, for which mean-value rule evaluation and monotonicity tests are applied again. In this **iterative range calculation**, the

Table 1. Monotonicity test in VALENCIA-IVP**.**

	$\inf\left(\frac{\partial r_i}{\partial z_j}\right) > 0$	$\sup\left(\frac{\partial r_i}{\partial z_j}\right) < 0$
$\inf\left\{ r_i(z)\Big\vert_{z_j=\xi_j} \right\}$	$\xi_j = \underline{z}_j$	$\xi_j = \overline{z}_j$
$\sup\left\{ r_i(z)\Big\vert_{z_j=\xi_j} \right\}$	$\xi_j = \overline{z}_j$	$\xi_j = \underline{z}_j$

component j_i^* of $[z]$ defined by

$$j_i^* = \underset{j=1,\ldots,n+1}{\arg\max} \left\{ \operatorname{diag} \left\{ \operatorname{diam} \left\{ \frac{\partial r_i}{\partial z}\bigg|_{z=[z]} \right\} \right\} \cdot \operatorname{diam}\{[z]\} \right\}, \tag{20}$$

is split at its interval midpoint [1,6]. For this component, the maximum reduction of overestimation is expected. Splitting is continued with the input intervals which lead to the smallest infimum/ largest supremum to improve the bounds of r_i until a user-defined number of subintervals is reached or until r_i is monotonic for all arguments. Finally, the *union* of all subintervals for r_i is used as the improved enclosure [1]. All partial derivatives required for that purpose are determined by algorithmic differentiation using FADBAD.

4 Consistency Test in VALENCIA-IVP for Reduction of Overestimation

In this Section, the consistency test used in VALENCIA-IVP is described for a time interval $t \in [t_i\ ;\ t_{i+N_c}]$, where $N_c > 1$ is the number of forward evaluation steps after which the consistency test is performed. This test aims at reducing overestimation that appears due to the replacement of the exact (often complexly shaped) regions in the state-space by axis-parallel interval boxes. It detects and eliminates subintervals originating from overestimation.

4.1 Basic Framework for the Consistency Test in VALENCIA-IVP

Step 1: Forward Evaluation from t_i to t_{i+N_c}. First, a forward evaluation of the set of state equations is performed using the basic version of VALENCIA-IVP to obtain the interval enclosure $[x(t_{i+N_c})]$. For an efficient consistency test, sufficient overestimation has to be present in this enclosure. Therefore, it is not necessary to perform the consistency test for each t_i. Typical values are $10 \leq N_c \leq 1000$.

Step 2: Subdivision of Interval Boxes at $t = t_{i+N_c}$. After the forward evaluation, the resulting interval enclosure is split into subintervals $[\tilde{x}_j(t_{i+N_c})] \subset [x(t_{i+N_c})]$, $j = 1, \ldots, L_c$, by the splitting strategies summarized in Subsection 4.2. The user-defined maximum number of subintervals is denoted by L_c. Typical values of this parameter suitable for most systems are $20 \leq L_c \leq 1000$.

Step 3: Backward Evaluation from t_{i+N_c} to t_i. At this stage, backward integration of each subinterval determined in **Step 2** is performed using the basic version of VALENCIA-IVP. During backward integration, the intersection with the guaranteed state enclosures determined by the forward evaluation is computed. This leads to tighter state enclosures due to reduced overestimation and increases the probability to detect inconsistent subintervals.

Step 4: Elimination of Inconsistent Subintervals. For each subinterval $[\tilde{x}_j(t_{i+N_c})]$ from the **Steps 2 and 3**, three different cases have to be distinguished, see Fig. 1.

Case (A): Subintervals which certainly originate from overestimation. Such subintervals are deleted. The intersection of the result of the backward integration of these subintervals with the state enclosure of the forward evaluation is empty in at least one component of the state vector for at least one point of time in the time interval $[t_i\ ;\ t_{i+N_c}]$.

Case (B): Subintervals which are consistent with $[x(t_i)]$. These subintervals belong to the solution. Their backward integration leads to time responses which are completely included in the result of the forward integration for all $t \in [t_i\ ;\ t_{i+N_c}]$.

Case (C): All other subintervals. Further splitting is required to check consistency.

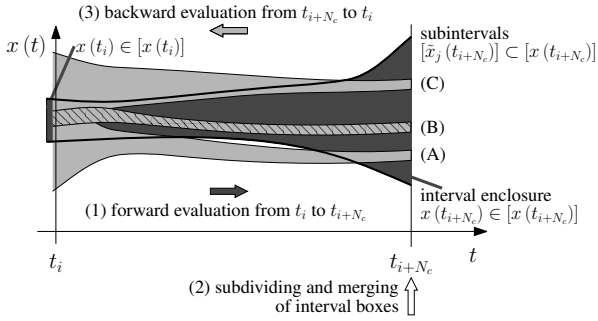


Figure 1. Consistency test in VALENCIA-IVP**.**

4.2 Splitting Procedure

The splitting procedure relies on two different selection criteria. The first one is to prefer large subintervals near the boundary of the set computed in the forward evaluation for $t = t_{i+N_c}$. The alternative is to select subintervals with small relative overlapping with the result of the forward step over the *complete* time interval $[t_i\ ;\ t_{i+N_c}]$. In both cases, only subintervals which are larger than some user-defined pseudo volume (product of the diameters of all components of an interval vector) are split.

4.3 Merging Strategy and Propagation of Subintervals

The subsequent forward evaluations are performed separately for each subinterval which is not inconsistent. To prevent an unlimited growth of the necessary computational effort, application of merging strategies is inevitable. Subintervals are replaced by their interval hull if this replacement leads to small additional overestimation. For that purpose, the pseudo volume of the *exact* union of two interval boxes $\left[x^{(\alpha)}\right]$ and $\left[x^{(\beta)}\right]$ given by

$$V_e = \mathrm{vol}\left(\left[x^{(\alpha)}\right]\right) + \mathrm{vol}\left(\left[x^{(\beta)}\right]\right) - \mathrm{vol}\left(\left[x^{(\alpha)}\right] \cap \left[x^{(\beta)}\right]\right) \tag{21}$$

and the pseudo volume

$$V_h = \mathrm{vol}\left(\left[x^{(\alpha)}\right] \cup \left[x^{(\beta)}\right]\right) \tag{22}$$

of their common interval *hull* are compared. The interval boxes $[x_\alpha]$ and $[x_\beta]$ are replaced by their interval hull if

$$\frac{V_h - V_e}{V_e} \cdot 100\% \leq \delta_{hull,limit} \ . \tag{23}$$

Typical values for the limit value in (23) are $\delta_{hull,limit} \in [0.1\% \ ; \ 5\%]$, see also [6]. Application of this merging routine limits the number of interval boxes and, thus, the computational effort. Besides, a user-defined simulation quality can be achieved.

5 Simulation Results

In the following, simulation results for COSY VI, VALENCIA-IVP, and VNODE are compared using two practically relevant examples. All solvers were compiled and all simulations were performed on a standard PC (Intel Pentium IV, 3.0 GHz, 1 GB RAM) under CYGWIN. Saving of intermediate results was switched off for each solver.

5.1 Application 1: Double Pendulum

The first application for the validated ODE solvers is a double pendulum (Fig. 2) with two massless arms $l_1 = l_2 = 1\,\mathrm{m}$ and two point masses $m_1 = m_2 = 1\,\mathrm{kg}$. The dynamical system model is summarized in (24) with the state vector $\psi(t) = \begin{bmatrix}\psi_1 & \psi_2 & \psi_3 & \psi_4\end{bmatrix}^T$. The initial angles ψ_1^0 and ψ_2^0 (in rad) as well as the initial angular velocities $\psi_3^0 = \dot{\psi}_1^0$ and $\psi_4^0 = \dot{\psi}_2^0$ (in $\frac{\mathrm{rad}}{\mathrm{s}}$) are given according to

$$\inf\left(\left[\psi^0\right]\right) = \begin{bmatrix}0.99\frac{3\pi}{4} & -\frac{11\pi}{20} & 0.43 & 0.67\end{bmatrix}^T$$
$$\text{and} \quad \sup\left(\left[\psi^0\right]\right) = \begin{bmatrix}1.01\frac{3\pi}{4} & -\frac{11\pi}{20} & 0.43 & 0.67\end{bmatrix}^T \ .$$

Even without monotonicity tests and iterative range computations (denoted by AIM), VALENCIA-IVP leads to tighter bounds than VNODE. For COSY VI with constant step sizes and QR preconditioning, the step size should be reduced below $0.002\,\mathrm{s}$ to obtain significantly better bounds than the ones computed with VALENCIA-IVP. However, this leads to high computing times, see Tabs. 2 and 3. In Tab. 2, the comparison has been performed with a constant step size $h = 0.002$ for all solvers. QR preconditioning and a series expansion of order 12 were used in VNODE and COSY VI (with identical orders for expansion in time and initial states). The effects of other fixed and variable step sizes are summarized in Tab. 3. Variable step sizes do not lead to much tighter bounds; only CPU times are reduced for both COSY VI and VNODE.

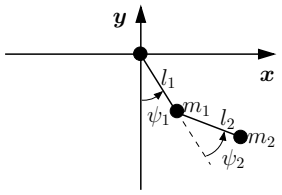


Figure 2. A double pendulum.

Table 2. Comparison for the double pendulum ($h = 0.002$).

	break-down time	reduction factor at $t = 0.5$	CPU time over $[0; 0.5]$
VNODE (QR)	0.5221	1.0	9.62 s
COSY VI (QR)	0.6300	$1.50 \cdot 10^4$	1.00 h
VALENCIA: MVR	0.5600	$1.32 \cdot 10^2$	1.31 s
VALENCIA: AIM	0.5920	$7.65 \cdot 10^2$	14.0 s

The investigation of (small) constant step sizes is especially important for model based state and parameter estimation. In these applications, dynamical interval observers relying on the integration of ODEs are used to perform prediction steps between the points of time at which measured data are available [13]. The grid is therefore predefined by the sampling frequency of the measurement device and, thus, cannot be controlled by the ODE solver.

In order to reduce the computing time in VALENCIA-IVP further, we plan to include strategies for automatic step size control for cases that do not require the step size to be constant. The comparison of computing times for VNODE and COSY VI clearly shows the benefits of step size control which are expected to be the same for VALENCIA-IVP.

In contrast to VNODE, QR preconditioning in COSY VI improves the state enclosures and break-down times only for relatively small and variable step sizes in this example. Here, preconditioning even leads to a considerable performance decrease for larger step sizes, see Tab. 3. Preconditioning is not essential in this case, since the poly-

$$\dot{\psi}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & m_1 l_1 + m_2 (l_1 + l_2 \cos(\psi_2)) & m_2 l_2 \cos(\psi_2) \\ 0 & 0 & m_2 (l_1 \cos(\psi_2) + l_2) & m_2 l_2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \psi_3 \\ \psi_4 \\ -g(m_1 + m_2)\sin(\psi_1) + m_2 l_2 \sin(\psi_2)(\psi_3 + \psi_4)^2 \\ -g m_2 \sin(\psi_1 + \psi_2) - m_2 l_1 \sin(\psi_2)\psi_3^2 \end{bmatrix} \quad (24)$$

nomial parts of Taylor models are a means to represent not axis-parallel regions in the state space. However, QR preconditioning is often advantageous since it reduces overestimation caused by axis-parallel interval remainder bounds of Taylor models. At last, the simulation with COSY VI could be sped up further by adapting the orders of the series expansions in the initial states according to the relevance of their interval diameters[2].

Table 3. Comparison for the double pendulum (cont'd).

VALENCIA-IVP: Version 0.92_2d (December 19, 2006), MVR | AIM					
step size, preconditioning			break-down time		CPU time over [0;0.5]
0.02			0.4400	0.4400	0.18 s (*) | 3.00 s (*)
0.002			0.5600	0.5920	1.31 s | 14.0 s
0.0002			0.5872	0.6124	11.0 s | 101 s
VNODE-2.0, constant/ variable step sizes, order 12					
0.02	with QR	tol=1e-16	0.4857		1.02 s (*)
0.002	with QR	tol=1e-16	0.5221		9.62 s
0.0002	with QR	tol=1e-16	0.5284		97.1 s
variable	with QR	tol=1e-16	0.5330		6.38 s
COSY VI: Version of March 22, 2004; COSY 9 (2004) constant/ variable step sizes, order 12 for expansion in states and time					
0.02	without QR	tol=1e-16	0.4800		24.6 s (*)
0.002	without QR	tol=1e-16	1.1320		3 min
0.0002	without QR	tol=1e-16	1.1212		31 min
variable	without QR	tol=1e-16	1.1212		31 min
variable	without QR	tol=1e-1	0.7882		27.36 s
0.02	with QR	tol=1e-16	0.3800		7 min (*)
0.002	with QR	tol=1e-16	0.6300		1 hour
0.0002	with QR	tol=1e-16	2.0610		7 hrs 59 min
variable	with QR	tol=1e-16	2.0610		7 hrs 59 min
variable	with QR	tol=1e-1	2.2905		16 min

(*) computing times until break-down of the corresponding solver
COSY VI: variable step size $h \in [0.0002\ ;\ 0.02]$, initial step size 0.02

The consistency test in VALENCIA-IVP with $N_c = 50$ and $L_c = 50$ leads to significant reduction of overestimation and a break-down time of approx. 0.9915. Especially, it can prevent growth of the enclosures over simulation time. For example, in the region marked by (*) in Fig. 3 (which is enlarged in the lower part), the consistency test in VALENCIA-IVP helps to compute interval bounds which are contracting over simulation time with a quality similar to COSY VI.

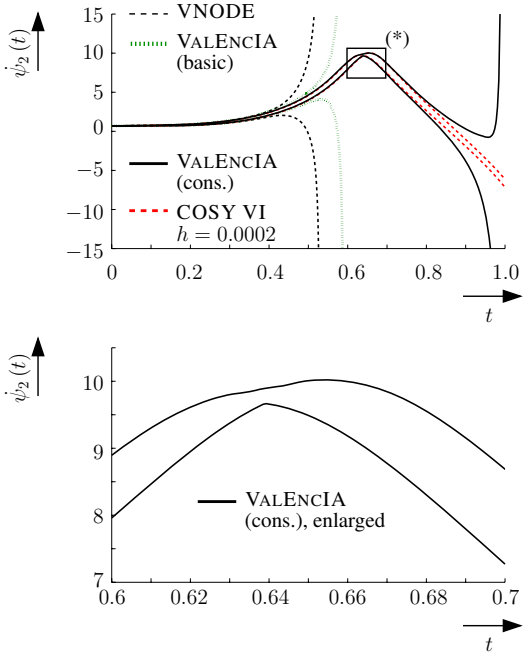


Figure 3. Enclosures for the angular velocity $\dot{\psi}_2(t)$ of the double pendulum ($h = 0.0002$).

5.2 Application 2: Subsystem Model of Biological Wastewater Treatment

The second application is a subsystem model of biological wastewater treatment. In this subsystem, the concentration S of biodegradable organic substrate is reduced by heterotrophic bacteria with concentration X under external oxygen supply with the flow rate u_{O2}. The concentration of dissolved oxygen in the aeration tank is denoted by S_O. The bacteria concentration in the settler (modeled as a perfect separator of sludge and purified water) is denoted by X_{Set}. A portion of the activated sludge is fed back into the aeration tank (flow rate Q_{RS} of return sludge); the excess sludge Q_{EX} is removed from the process, see Fig 4.

[2]The recently released version of COSY VI (version of Nov. 10, 2006; COSY 9 of August, 2006) which supports the selection of different orders for the series expansion in the components of the initial state vector will be studied in future comparisons. All state equations used for the comparison of the selected solvers are available via `http://www.valencia-ivp.com`.

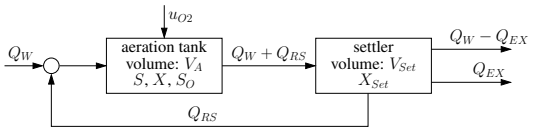


Figure 4. Block diagram of a simplified biological wastewater treatment process.

Simplification of the ASM1 (Activated Sludge Model No. 1 of the International Water Association) leads to a set of four nonlinear ordinary differential equations

$$\begin{aligned}
\dot{S} =& \frac{Q_W}{V_A}\left(S_W - S\right) - \mu\left(S, S_O\right)\frac{1}{Y}X \\
\dot{X} =& -\frac{Q_W}{V_A}X + \frac{Q_{RS}}{V_A}\left(X_{Set} - X\right) \\
&+ \left(\mu\left(S, S_O\right) - b\right)X \\
\dot{S}_O =& \frac{Q_W}{V_A}\left(S_{OW} - S_O\right) - \mu\left(S, S_O\right)\frac{1-Y}{Y}X \\
&+ \frac{\rho_{O2}}{V_A}\left(1 - \frac{S_O}{S_{O,sat}}\right)u_{O2} \\
\dot{X}_{Set} =& \frac{Q_W + Q_{RS}}{V_{Set}}X - \frac{Q_{EX} + Q_{RS}}{V_{Set}}X_{Set} \ ,
\end{aligned} \tag{25}$$

where the nonlinear growth rate of substrate consuming bacteria is modeled by the Monod kinetics

$$\mu\left(S, S_O\right) = \hat{\mu}_H \frac{S}{S + K_S}\frac{S_O}{S_O + K_{OS}} \ . \tag{26}$$

A more detailed description and a complete list of all parameters can be found in [12].

In practice, most parameters are uncertain due to variations of amount and composition of the inflow into the aeration tank as well as various weather conditions. Especially, the temperature of the wastewater has a strong influence on the maximum specific growth rate $\hat{\mu}_H$ and on the decay rate b of the biomass. In the following, $\hat{\mu}_H$ is chosen as the only uncertain parameter with $\hat{\mu}_H \in [0.9\ ;\ 1.1]\,\hat{\mu}_{H,nom}$. Aside from bounds of these parameters, also uncertainties of their variation rates can often be specified according to (2)–(4). In VALENCIA-IVP, this case can be handled by introduction of additional state variables for the parameters and limitation of their range *after* computation of the error bounds $[R(t)]$ by intersection with the limit values which are expressed in terms of bounds for the approximation error intervals. VNODE can be extended analogously. In this case, the intersection with the known bounds of the parameter intervals is necessary after calculation of the state enclosures in **Stage Two**. However, for COSY VI, this intersection cannot be performed directly since the current version of this solver does not provide a possibility to reformulate the result of this intersection as a Taylor model without loss of the information stored in the Taylor model before the limitation. Therefore, the following comparison is restricted to the *time-invariant* case.

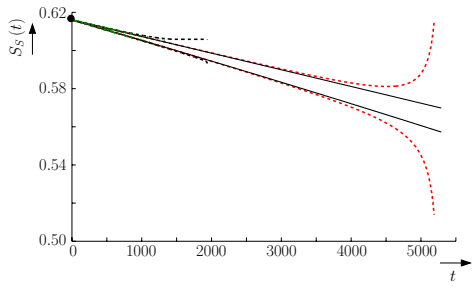
Table 4. Comparison of the ODE solvers for the wastewater treatment process.

COSY VI: Variable step size $h \in [0.02\ ;\ 100]$		
order (t, x^0)	CPU time $(t = 500)$	break-down time
5, 5	15.52 s	5935
10, 10	162.32 s	5188
17, 5	84.45 s	6168
COSY VI: Constant step size $h = 0.25$		
10, 10	2353.00 s	≈ 5100
VNODE: Variable step size		
order	CPU time $(t = 500)$	break-down time
5	0.797 s	1800
10	0.344 s	811
17	0.469 s	543
VNODE: Constant step size $h = 0.25$		
10	18.7 s	1939
VALENCIA-IVP: Constant step size, without consistency test		
step size h	CPU-time $(t = 500)$	break-down time
0.25	67 s	885
0.025	478 s	1335
VALENCIA-IVP: Constant step size, with consistency test		
0.25	5680 s	> 6000

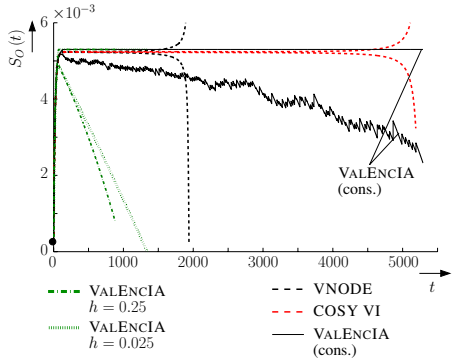
The results for the wastewater treatment process are summarized in Tab. 4. It can be observed that for orders ≥ 5 of the series expansion in time, break-down times of at least $5,000$ s can be achieved for COSY VI with variable step sizes $h \in [0.02\ ;\ 100]$ s. Although changes of the chosen orders do not have much influence on the quality of the state enclosures, the required computing times vary significantly (see Tab. 4). For suitable constant step sizes, only the computing time is increased, without much influence on the simulation quality. For VNODE, constant step sizes lead to tighter bounds compared to a simulation with the same order and a variable step size. Using higher orders of the series expansion leads to worse results because of increased overestimation in the computation of Taylor coefficients due to the rational terms in the ODEs.

For VALENCIA-IVP (approximate solutions calculated by an explicit Euler method with the step sizes defined in Tab. 4) without consistency test, break-down times are comparable to VNODE with variable step sizes. However, the bounds for the rapidly changing concentration S_O are worse for VALENCIA-IVP (see Fig. 5). This shows that application of the consistency test is almost inevitable for stiff systems. To reduce the problems caused by the stiffness of the system, the consistency test was performed with $N_c = 100$, $L_c = 100$, and $h = 0.25$. Now, VALENCIA-IVP produces the best bounds of all selected solvers for S, X, and X_{Set}. The applied monotonicity test and iterative range computation are especially efficient for reduction of the dependency problem for rational terms such as the Monod kinetics (26) without the necessity for symbolic simplification of

the ODEs. Future optimizations of the consistency test will aim at improved implementations (such as automatic selection of N_c and L_c) and further strategies to reduce computing times, especially since the pseudo volume of the enclosures at $t = 500$ could be reduced by a factor of 470 compared to the basic version of VALENCIA-IVP.



(a) Substrate concentration S.



(b) Oxygen concentration S_O.

Figure 5. State enclosures for the wastewater treatment process.

6 Conclusions and Outlook on Future Work

In this paper, a new consistency test for the validated ODE solver VALENCIA-IVP was introduced. Using two applications, it was shown that the basic version of this solver can produce simulation results which are comparable to the ones obtained by VNODE. For several applications, VALENCIA-IVP needs less CPU time than COSY VI. It was demonstrated that the new consistency test for elimination of state intervals which result from overestimation is a promising approach for simulations of higher-dimensional problems. The basic idea of this test can in general be applied to any other validated ODE solver. Especially, an implementation in fast validated solvers such as VNODE will be an interesting topic for future research. Additionally, improvements of the consistency test by implicit integration techniques as well as automatic selection of the number of subintervals and the time horizon for backward integration will be considered to reduce computing time. Another direction for future research will be the extension of VALENCIA-IVP towards systems of differential-algebraic equations.

References

[1] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SMARTMOBILE: Improvement of Performance by VALENCIA-IVP. In *Proc. of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice*, Lecture Notes in Computer Science, 2006. In print.

[2] C. Bendsten and O. Stauning. FADBAD, a Flexible C++ Package for Automatic Differentiation Using the Forward and Backward Methods. Technical Report 1996-x5-94, Technical University of Denmark, Lyngby, 1996.

[3] C. Bendsten and O. Stauning. TADIFF, a Flexible C++ Package for Automatic Differentiation Using Taylor Series. Technical Report 1997-x5-94, Technical University of Denmark, Lyngby, 1997.

[4] M. Berz and K. Makino. Performance of Taylor Model Methods for Validated Integration of ODEs. *Lecture Notes in Computer Science*, 3732:65–74, 2005.

[5] O. Knüppel. PROFIL/BIAS — A Fast Interval Library. *Computing*, 53:277–287, 1994.

[6] I. Krasnochtanova. Optimized Interval Algorithms for Simulation and Controller Design for Nonlinear Uncertain Systems Applied to Processes in Biological Wastewater Treatment, 2005. Master Thesis, University of Ulm.

[7] W. Kühn. Rigorous Error Bounds for the Initial Value Problem Based on Defect Estimation. Technical report, 1999. `http://www.decatur.de/wolfgang/papers/index.html`.

[8] R. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen*. PhD thesis, Universität Karlsruhe, 1988.

[9] K. Makino and M. Berz. Suppression of the Wrapping Effect by Taylor Model-Based Verified Integrators: The Single Step. *International Journal of Pure and Applied Mathematics*, 2005. In print. Online.

[10] N. S. Nedialkov. *The Design and Implementation of an Object-Oriented Validated ODE Solver*. Kluwer Academic Publishers, 2002.

[11] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An Effective High-Order Interval Method for Validating Existence and Uniqueness of the Solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001.

[12] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Reduction of Overestimation in Interval Arithmetic Simulation of Biological Wastewater Treatment Processes. *Journal of Computational and Applied Mathematics*, 199(2):207–212, 2007.

[13] A. Rauh, M. Kletting, and E. P. Hofer. Model-Based State and Parameter Estimation for Micro-Mechatronic Systems with Interval Bounded Uncertainties. In A. Weckenmann, editor, *CD-Proc. of 10th CIRP Intl. Conference on Computer Aided Tolerancing, Erlangen, Germany, 2007*, Reports from the Chair Quality Management and Manufacturing Metrology, QFM Report 16. Shaker Verlag, Aachen, 2007.

Interval Techniques for Design of Optimal and Robust Control Strategies

Andreas Rauh, Johanna Minisini, and Eberhard P. Hofer
Institute of Measurement, Control, and Microtechnology
University of Ulm
D-89069 Ulm, Germany
{Andreas.Rauh, Johanna.Minisini, EP.Hofer}@uni-ulm.de

Abstract

In this paper, an interval arithmetic optimization procedure for both discrete-time and continuous-time systems is presented. Besides computation of control strategies for systems with nominal parameters, robustness requirements for systems with interval bounded uncertainties are considered. Considering these uncertainties, control laws are obtained which directly take into account the influence of disturbances and deviations of system parameters from their nominal values. Compared to Bellman's discrete dynamic programming, errors resulting from gridding of state and control variable intervals as well as errors due to rounding to nearest grid points are avoided. Furthermore, the influence of time discretization errors is taken into account by validated integration of continuous-time state equations. Optimization results for a simplified model of a mechanical positioning system with switchings between models for both static and sliding friction demonstrate the efficiency of the suggested approach and its applicability to processes with state-dependent switching characteristics.

1 Introduction

In recent years, different optimization techniques for dynamical systems have been developed for both discrete-time and continuous-time systems. The most important optimization procedures for *continuous-time* systems described by ordinary differential equations (ODEs) are based on Pontryagin's maximum principle [14] and the Hamilton-Jacobi-Bellman equation [5, 8]. The maximum principle leads to a boundary value problem for a set of ODEs while the Hamilton-Jacobi-Bellman equation is a nonlinear partial differential equation. In both cases, it is necessary to apply numerical techniques in order to determine optimal solutions for nonlinear real-world models describing technical applications.

For *discrete-time* systems, Bellman's dynamic programming is the most universal approach which — at least theoretically — leads to globally optimal solutions [2, 3]. However, this approach suffers from the so-called curse of dimensionality which means that for increasing dimensions of the state-space the computational effort grows exponentially since most implementations rely on gridding of the admissible range of both state and control variables.

Originally, this procedure has been developed for discrete-time systems. In order to apply dynamic programming techniques to nonlinear continuous-time processes, time discretization is required. If the resulting discretization error is neglected, often considerable deviations from the original continuous-time systems arise. Hence, such phenomena have to be taken into account during optimization. In this contribution, interval arithmetic methods [6, 10] are applied to enclose the arising discretization errors by guaranteed interval bounds. Additionally, uncertainties of system parameters are considered simultaneously to analyze the robustness of the solutions and to determine control strategies which make the controlled system robust against parameter variations.

To reduce problems caused by the curse of dimensionality, intelligent strategies to search for the global optimum of the performance index have to be applied. These strategies include disregarding all control sequences which are either not admissible due to violation of restrictions of the state variables or which are not optimal with respect to the performance index. Control sequences which are not optimal are eliminated already during the optimization process in order to make sure that the search for globally optimal control strategies is only performed for control variable intervals in which the optimum is included. This reduces the computational burden significantly but does not eliminate the curse of dimensionality completely. In contrast to other implementations of dynamic programming, gridding is avoided. Instead, the suggested approach leads to an adaptive refinement of the control variable intervals near the optimum of the performance index.

Intermediate solutions which are constant for several

subsequent time steps are determined to reduce the computational effort by elimination of control strategies which are not optimal in early stages of the optimization. In the case of multiple control variables, further suboptimal solutions are obtained by selection of a small number of control variables for the optimization while all other control variables are assumed to be fixed.

In Section 2, a detailed formulation of the considered optimization problems is given for discrete-time and continuous-time systems. The recently developed interval arithmetic optimization routine is introduced in Section 3. In Section 4, possibilities for combination of the optimization technique with classical controller design are discussed to reduce the controlled systems' sensitivity w.r.t. parameter variations. For demonstration purposes, optimal control of a simplified continuous-time mechanical positioning system with state-dependent switchings between different dynamical models which consider viscous friction together with Coulomb friction [17] is discussed in Section 5. Finally, in Section 6, an outlook on future research is given.

2 Optimal and Robust Control of Dynamical Systems

In this Section, the problem of optimal and robust control of both continuous-time and discrete-time processes is formulated. In both cases, transfer of the initial state vector to the desired final state vector should be performed such that a predefined performance index is minimized by choosing an admissible control strategy. In general, the two different problems of parameter optimization on the one hand and structure optimization on the other hand can be distinguished.

In the parameter optimization problem, parameters of a controller with a fixed structure, e.g. a P-, PI-, PID-, or linear state controller, have to be determined. In contrast, the result of the structure optimization problem is an optimal open- or closed-loop control strategy which is determined without making any assumptions about the structure of the controller. Although the presented optimization routines can be applied to both problems, this paper focuses on structure optimization.

2.1 Optimal Control of Discrete- and Continuous-Time Processes

For discrete-time dynamical systems

$$x_{k+1} = g_k\left(x_k, p_k, u_k, k\right) \quad , \tag{1}$$

$g_k : D \mapsto \mathbb{R}^{n_x}, D \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_u} \times \mathbb{R}^1$, with the state vector $x_k \in \mathbb{R}^{n_x}$ and the vector $p_k \in \mathbb{R}^{n_p}$ of system parameters, an initial state x_0 should be transferred into a desired final state $x\left(k_{max}\right)$, such that the performance index

$$J = \sum_{k=0}^{k_{max}} g_{J,k}\left(x_k, p_k, u_k, k\right) \tag{2}$$

is minimized by calculation of an admissible control sequence $u_k \in \mathbb{R}^{n_u}$. Analogously, continuous-time processes described by the state equations

$$\dot{x}\left(t\right) = f\left(x\left(t\right), p\left(t\right), u\left(t\right), t\right) \quad , \tag{3}$$

$f : D \mapsto \mathbb{R}^{n_x}, D \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_u} \times \mathbb{R}^1$, with the state vector $x\left(t\right) \in \mathbb{R}^{n_x}$ and the vector $p\left(t\right) \in \mathbb{R}^{n_p}$ of system parameters can be considered. Again, an initial state x_0 should be transferred into a desired final state $x\left(T\right)$, such that the performance index

$$\begin{aligned} J = & f_T\left(x\left(T\right), p\left(T\right), T\right) \\ & + \int_0^T f_0\left(x\left(t\right), p\left(t\right), u\left(t\right), t\right) dt \end{aligned} \tag{4}$$

is minimized. For that purpose, the control law $u\left(t\right) \in \mathbb{R}^{n_u}$ has to be determined.

Both optimization problems are summarized in the Figs. 1 and 2, where the time horizon is denoted by $k \in [0\ ;\ k_{max}]$ in the discrete-time case and by $t \in [0\ ;\ T]$ in the continuous-time case. Note that the terminal cost function $f_T\left(x\left(T\right), p\left(T\right), T\right)$ from definition (4) is included in the term $g_{J,k_{max}}$ in (2).

In addition to exactly known initial and final states, also free boundary conditions or states from a certain predefined region of initial or final states can be investigated. In many practical situations, uncertainties of the system parameters have to be taken into account during the optimization process. If guaranteed bounds for these values are known, the uncertain system parameters can be described by the intervals $p_k \in \left[\underline{p}_k\ ;\ \overline{p}_k\right]$ and $p\left(t\right) \in \left[\underline{p}\left(t\right)\ ;\ \overline{p}\left(t\right)\right]$, resp., which represent the maximum possible tolerances. Furthermore, parameters which are not constant over the complete time horizon of the optimization process have to be modeled by additional discrete-time state equations

$$p_{k+1} = p_k + \Delta p_k \tag{5}$$

with the bounded variation rates

$$\Delta p_k \in \left[\underline{\Delta p}_k\ ;\ \overline{\Delta p}_k\right] \tag{6}$$

or by additional ordinary differential equations

$$\dot{p}\left(t\right) = \Delta p\left(t\right) \tag{7}$$

with

$$\Delta p\left(t\right) \in \left[\underline{\Delta p}\left(t\right)\ ;\ \overline{\Delta p}\left(t\right)\right] \tag{8}$$

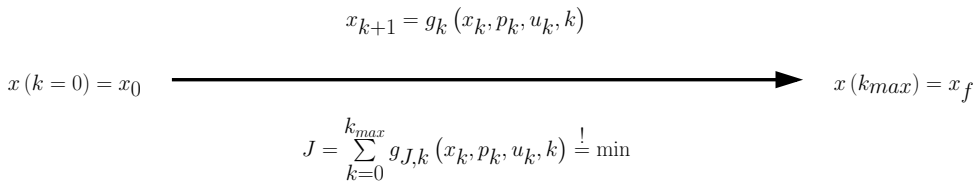


Figure 1. Optimization problem for discrete-time dynamical systems.

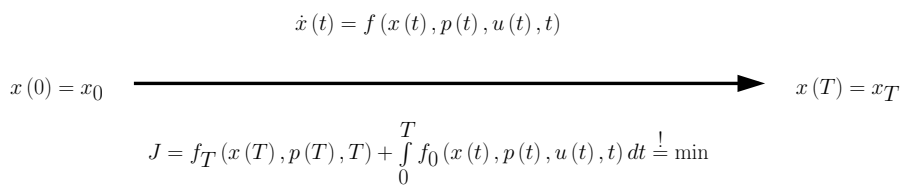


Figure 2. Optimization problem for continuous-time dynamical systems.

in the case of continuous-time problems.

During the optimization procedure, limitations of all control variables u_k and $u\left(t\right)$ have to be considered. In this paper, it is assumed, that the vector of control variables is bounded by the intervals $u_k \in [\underline{u}_k \; ; \; \overline{u}_k]$ and $u\left(t\right) \in [\underline{u}\left(t\right) \; ; \; \overline{u}\left(t\right)]$, resp. These bounds do not need to be constant over the prescribed time horizon.

2.2 Robustness Specifications

As already pointed out, one of the main properties of the presented optimization procedure is that it can directly deal with interval uncertainties of the system parameters. Hence, robustness of the controlled system w.r.t. parameter variations as well as optimality of a control sequence under consideration of parameter uncertainties have to be defined.

Robustness specifications are assumed to be given by worst-case bounds of all system states which must not be violated during transfer of the initial state into the desired final state using a control sequence which is completely inside its bounds for all points of time [16]. Furthermore, only the definition of either free final states or bounded regions of admissible final states makes sense in the case of uncertain parameters, since, in general, one common control sequence for all possible values of the uncertain parameters will not be able to eliminate the influence of parameter uncertainties completely.

In the case of simultaneous consideration of the abovementioned bounds of the state variables and an optimization criterion as defined in (2) and (4), a control sequence is said to be optimal if it does not violate any of the specified bounds; at the same time the optimal control sequence must lead to the *smallest possible upper bound* of the performance index if the maximum influence of the uncertainties is investigated.

The interval arithmetic optimization algorithm presented in the following aims at calculating control sequences within the prescribed bounds of the control variable intervals. The resulting trajectories of the state variables have to be included completely in the regions of admissible states for each possible parameter value.

3 Interval Arithmetic Optimization Algorithm

The interval arithmetic optimization algorithm presented in this Section is an extension of a procedure presented by the authors in [15]. In addition to the original version, the new version can not only deal with discrete-time systems with nominal parameters; both discrete- and continuous-time dynamical models including parameter uncertainties as well as limitations of state variables representing time-domain robustness specifications can be handled. For continuous-time processes, a piecewise constant control law with a predefined sampling time, which is independent of the step sizes used by the underlying validated ODE solvers, is computed.

Step OPT 1 Based on evaluation from the final to the initial point of time, i.e., from $k = k_{max}$ to $k = 0$, or $t = T$ to $t = 0$, resp., enclosures of all states are determined by backward evaluation of the state equations.

For discrete-time systems, the state equation (1) is solved for the state x_k under the assumption that an interval enclosure for x_{k+1} is known. If an analytical solution

$$x_k = \tilde{g}_k\left(x_{k+1}, p_k, u_k, k\right) \tag{9}$$

does not exist, interval Newton methods are used instead. Analogously, for continuous-time systems backward integration of the state equations

$$\dot{x}\left(t\right) = f\left(x\left(t\right), p\left(t\right), u\left(t\right), t\right) \tag{10}$$

is performed for given $x(T)$ until $t = 0$ is reached. In both cases, the computation is performed for the known interval bounds of the uncertain parameters and the limited range of the control variable intervals.

Simultaneously, interval enclosures of the corresponding performance indices

$$\begin{aligned} J_k &= J_{k+1} + g_{J,k}\left(x_k, p_k, u_k, k\right) \\ &= J_{k+1} + g_{J,k}\left(\tilde{f}_k\left(x_{k+1}, p_k, u_k, k\right), p_k, u_k, k\right) \end{aligned} \tag{11}$$

and

$$\begin{aligned} J_t =& f_T\left(x(T), p(T), T\right) \\ &+ \int_t^T f_0\left(x(\tau), p(\tau), u(\tau), \tau\right) d\tau \end{aligned} \tag{12}$$

are determined, where the terminal cost functions are denoted by $J_{k_{max}} = g_{J,k_{max}}\left(x_{k_{max}}, p_{k_{max}}, u_{k_{max}}, k_{max}\right)$ and $J_T = f_T\left(x(T), p(T), T\right)$. The costs for transfer from the time step k to k_{max} (or from t to T) are denoted by J_k (or J_t). The general idea of the presented optimization algorithm is the minimization of the performance index J_0 by repeated splitting of the control variable intervals. This procedure leads to an approximation of the optimal control sequences $\{u_k^*\}$ and $u^*(t)$ as well as the optimal trajectories $\{x_k^*\}$ and $x^*(t)$ for all $k \in [0\ ;\ k_{max}]$ and $t \in [0\ ;\ T]$.

Using interval techniques for backward evaluation of the state equations, validated enclosures of the regions of attraction are calculated which can be transferred into the desired final state under consideration of all possible values of the uncertain system parameters. Hence, the intervals $[J_k]$ and $[J_t]$ represent worst-case bounds of the range of the performance index for all control variables from the admissible range $[\underline{u}\ ;\ \overline{u}]$. Thus, the influence of interval uncertainties is directly expressed in terms of the maximum variations of the system states and the performance index. Backward evaluation of the state equations is omitted if the final states are unbounded.

Step OPT 2 Afterwards, the state equations are evaluated from the initial to the final point of time. Together with the results of **Step OPT 1**, candidates for control sequences are eliminated which do not allow to transfer a given initial state x_0 or region of initial states $[x_0]$ into the desired final state or region of final states. Additionally, control sequences are eliminated which are not optimal after comparison of the performance index intervals of several candidates.

Step OPT 3 The interval widths for $\{[x_k]\}$ and $[x(t)]$ as well as $\{[u_k]\}$ and $[u(t)]$ of candidates for optimal control strategies are reduced by repeated forward and backward evaluations in **Steps OPT 1** and **OPT 2** as long as further improvement is possible.

Step OPT 4 During the global optimization procedure two strategies have proven successful. First, sequences $\{[u_k]\}_{sup}$ or $\{[u(t)]\}_{sup}$ are selected such that the corresponding supremum of the performance index is smaller than the supremum of all other candidates. This leads to a fast reduction of the upper bound of the maximum necessary costs. For both discrete- and continuous-time systems, the control variable interval is split at the point of time $\tilde{k}$ according to

$$\tilde{k} = \underset{j=0,\ldots,k_{max}-1}{\arg\max} \left\{ \operatorname{diam}\left(\left[\frac{\partial J}{\partial u_j} \right] \right) \cdot \operatorname{diam}\left([u_j]\right) \right\} \tag{13}$$

with $\operatorname{diam}\left([u_j]\right) = \overline{u}_j - \underline{u}_j$, for which the largest reduction of the diameter of the performance index interval is expected. For continuous-time systems with piecewise constant control strategies $u(t_k)$ equation (13) is evaluated after discretization of the state equations with the same step size that is used for validated integration. Second, the sequences $\{[u_k]\}_{inf}$ or $\{[u(t)]\}_{inf}$ with the smallest infimum of the performance index are selected to improve the lower bound of the necessary costs. Here, $\tilde{k}$ is chosen again as in (13). Suboptimal control laws are obtained by performing the optimization under the assumption of control strategies which are *constant* for several subsequent time steps.

The optimization is stopped if the diameter of the performance index interval of the best known approximation of the optimal control sequence is smaller than a user-defined value and, at the same time, if the distance of the performance index from the estimate of its global infimum falls below another user-defined value.

Step OPT 5 Output of the best known approximation for the optimal control sequences $\{u_k^*\}$ and $u^*(t)$. Note that this approximation of the optimal control sequence is not ensured to be globally optimal. However, the results of the previous steps provide guaranteed lower and upper bounds of the costs which are necessary to perform the control task. By choosing appropriate stopping criteria for the optimization process, the user can make sure that the deviation between the approximation of the optimal control strategy and the global infimum of the performance index (assuming piecewise constant control strategies in both cases) is smaller that a prescribed value.

For the purpose of interval evaluation of *discrete-time* systems, recursive interval evaluation of the state equations, global optimization techniques, as well as consistency tests and state space transformations aiming at the reduction of overestimation are used. For *continuous-time* systems, arbitrary validated ODE solvers such as VNODE (using Taylor series expansions of the solution of an initial value problem IVP based on higher-order time derivatives of the ODE) [11, 12], COSY VI (Taylor model based solver

with additional series expansion in initial states) [4, 9], or VALENCIA-IVP [1] can be applied. Compared to the discrete-time case, guaranteed bounds for discretization errors determined by ODE solvers which rely on time discretization only represent an additional uncertain parameter in the optimization process.

4 Combination with Classical Controller Design

As it will be shown in the following application, closed-loop controllers can be applied successfully to reduce the influence of parameter variations on a controlled system's performance. In general, such controllers can be parameterized with the help of the previously described algorithm. For a given controller structure, either constant or time-varying controller parameters can be computed.

5 Application: Mechanical Positioning System with Friction

5.1 Modeling and Simulation of Dynamical Systems with State-Dependent Switching Characteristics

In this Section, the proposed optimization technique is applied to a system with state-dependent switchings between different dynamical models. This system represents a simplified model of a mechanical positioning unit with a sliding mass m as well as the given initial position $x_1(0) = 0$ and given initial velocity $x_2(0) = 0$. The state-dependent switching characteristic reflects the different dynamical behavior of the system in static and sliding friction. During the optimization, an optimal accelerating force $u(t) = F_a(t)$ should be determined such that the region of admissible final states $[x_1(T)] = [0.9\ ;\ 1.1]$, $[x_2(T)] = [-0.1\ ;\ 0.1]$ is reached for $T = 5$ for all uncertain parameters of the friction characteristic $F_f(x_2)$.

According to [17], where a general simulation procedure for systems with state-dependent switchings has been introduced, the positioning system is described by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{m}\left(F_a(t) - F_f(x_2)\right) \end{bmatrix} \qquad (14)$$

with the state vector $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. The friction characteristic $F_f(x_2)$ is modeled by three discrete model states $\mathcal{S} = \{S_1, S_2, S_3\}$, which are

- sliding friction for motion in "negative" (backward) direction ($= S_1$),
- static friction ($= S_2$), and
- sliding friction for motion in "positive" (forward) direction ($= S_3$).

In Fig. 3, the influence of interval parameters for the static friction coefficient $[F_s] := \left[\underline{F_s}\ ;\ \overline{F_s}\right]$ as well as the sliding friction coefficient $[\mu] := \left[\underline{\mu}\ ;\ \overline{\mu}\right]$ is depicted. The resulting sliding friction force is given by

$$F_f(x_2) = \begin{cases} -[F_s] + [\mu] \cdot x_2 & \text{for} \quad S_1 = \texttt{true} \\ +[F_s] + [\mu] \cdot x_2 & \text{for} \quad S_3 = \texttt{true} \end{cases} \qquad (15)$$

and the static friction force by

$$F_f(x_2) \in [F_s^{max}] := \left[-\overline{F_s}\ ;\ \overline{F_s}\right] \quad \text{for} \quad S_2 = \texttt{true}\ . \qquad (16)$$

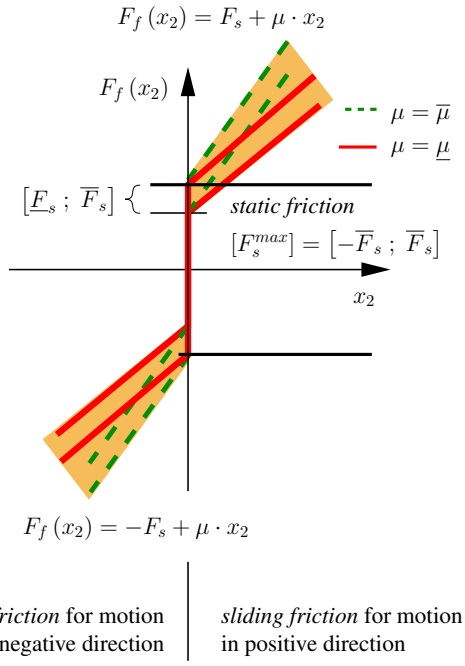


Figure 3. Friction characteristic with uncertain sliding friction coefficient $[\mu]$ and uncertain static friction coefficient $[F_s]$.

The state transition diagram in Fig. 4 displays the three model states together with the conditions for all possible transitions between these states. In the case of parameter uncertainties, several states $\mathcal{S}$ can be active simultaneously. The transition conditions T_i^j from state S_i to state S_j are expressed in terms of the state variables x_1 and x_2, the system parameters, and the control variable. In the transition diagram in Fig. 4, the union operator for n-dimensional interval vectors $[v]$ and $[w]$ is defined component-wise according to

$$[v] \cup [w] = [\min\{\underline{v}, \underline{w}\}\ ;\ \max\{\overline{v}, \overline{w}\}]\ . \qquad (17)$$

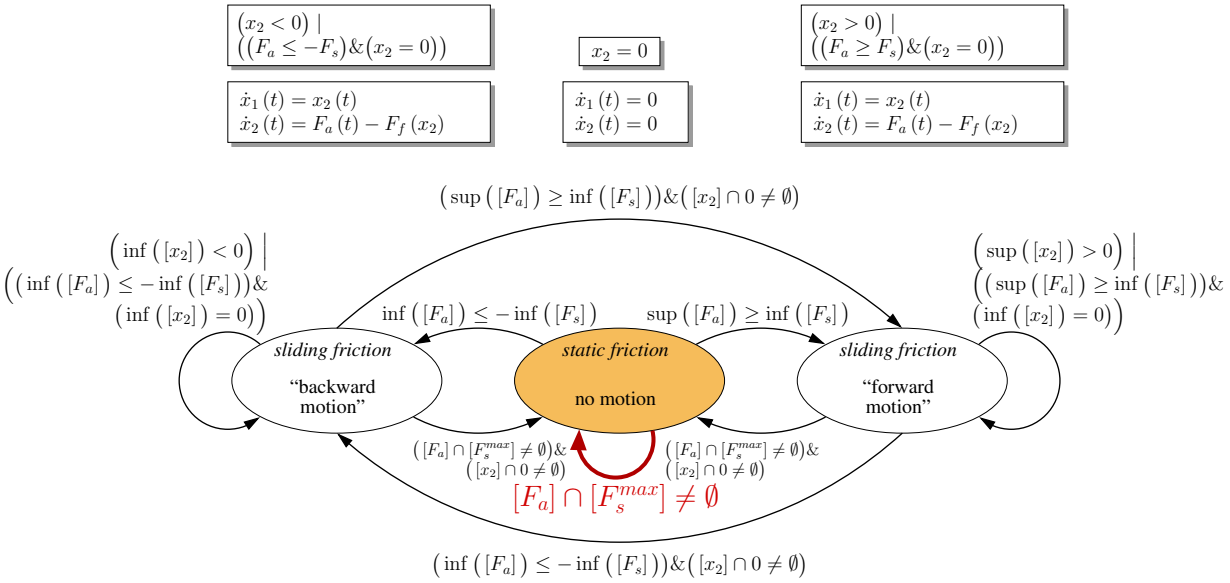


Figure 4. State-dependent switching model of friction characteristic (interval implementation).

Analogously, the intersection operator is defined as

$$[v] \cap [w] = \begin{cases} [\max\{\underline{v}, \underline{w}\} \ ; \ \min\{\overline{v}, \overline{w}\}] \\ \quad \text{if} \quad \max\{\underline{v}_i, \underline{w}_i\} \leq \min\{\overline{v}_i, \overline{w}_i\} \\ \quad\quad \forall i = 1, \ldots, n \\ \emptyset \quad \text{otherwise} \quad . \end{cases} \tag{18}$$

As an example, the highlighted transition condition

$$T_2^2 = [F_a] \cap [F_s^{max}] \neq \emptyset \tag{19}$$

in Fig. 4 indicates that the system remains in static friction if $|F_a| \leq \overline{F}_s$.

In the following, an extension of a Taylor series based integration algorithm is described which allows for computation of validated enclosures of all reachable states in the case of state-dependent switchings by detecting all points of time at which transition conditions are activated or at which one of the discrete model states is deactivated. Note that the following procedure can also be applied to any other validated ODE solver if appropriate adjustments are made. For further information about interval methods for the computation of guaranteed state enclosures for dynamical systems with state-dependent switchings, the reader is referred to [13, 18] and the references therein.

Step SIM 1 A bounding box $[B_{a,k}]$ of all reachable states in the time interval $[t_k \ ; \ t_{k+1}]$ is calculated by a Picard iteration for the state equation $f_a(x(t), p, u(t), t)$ which is the union of *all* models which are active at $t = t_k$, i.e.,

$$\mathbb{F}_{f_a} \supseteq \bigcup_{i \in \mathcal{I}_a} \mathbb{F}_{f_{S_i}} \ , \tag{20}$$

where

$$\mathcal{I}_a = \{ i \mid S_i = \texttt{true} \} \quad \text{for} \quad t = t_k \ . \tag{21}$$

In (20), $\mathbb{F}_{f_a}$ and $\mathbb{F}_{f_{S_i}}$ represent the *exact* value sets

$$\mathbb{F}_{f_a} = \{ y | y = f_a([x(t_k)], [p], [u(t_k)], t_k) \} \tag{22}$$

and

$$\mathbb{F}_{f_{S_i}} = \{ y | y = f_{S_i}([x(t_k)], [p], [u(t_k)], t_k) \} \tag{23}$$

of f_a and f_{S_i} under consideration of all interval arguments. The subscript a indicates that the state equation consists of the union of all *active* models.

Step SIM 2 If an additional transition condition is activated for one of the active models S_i, $i \in \mathcal{I}_a$, the bounding box $[B_{a,k}]$ in **Step SIM 1** has to be re-computed. I.e., if $\tilde{\mathcal{I}}_a \neq \mathcal{I}_a$, all additionally activated models have to be taken into account, where

$$\tilde{\mathcal{I}}_a = \mathcal{I}_a \cup \Big\{ j \mid \Big(T_i^j([B_{a,k}], u([t_{k,k+1}])) = \texttt{true} \Big) \cap \Big(i \in \mathcal{I}_a \Big) \Big\} \ . \tag{24}$$

Then, the modified state equation $f_a(x(t), p, u(t), t)$ is determined such that

$$\mathbb{F}_{f_a} \supseteq \bigcup_{i \in \tilde{\mathcal{I}}_a} \mathbb{F}_{f_{S_i}} \ , \tag{25}$$

with

$$\mathbb{F}_{f_a} = \{ y | y = f_a([B_{a,k}], [p], u([t_{k,k+1}]), [t_{k,k+1}]) \} ,$$

$$\mathbb{F}_{f_{S_i}} = \left\{ y | y = f_{S_i}\left([B_{a,k}], [p], u\left([t_{k,k+1}] \right), [t_{k,k+1}] \right) \right\},$$

and

$$[t_{k,k+1}] := [t_k \ ; \ t_{k+1}] \quad .$$

If $\tilde{\mathcal{I}}_a = \mathcal{I}_a$ in (24), evaluation is continued with **Step SIM 3**.

Step SIM 3 Interval enclosures of the state vector $[x_{k+1}]$ at t_{k+1} are computed by validated integration of $f_a\left(x\left(t\right), p, u\left(t\right), t\right)$ as defined in **Step SIM 2**. In Taylor series based methods, the analytical expression for $f_a\left(x\left(t\right), p, u\left(t\right), t\right)$ is also used to compute the required Taylor coefficients, see also [17].

Step SIM 4 All model states have to be deactivated which can no longer be active at t_{k+1}. Afterwards the simulation is continued with **Step SIM 1** for the next time interval $[t_{k+1,k+2}] := [t_{k+1} \ ; \ t_{k+2}]$.

5.2 Optimization Results

In this Subsection, optimization of the dynamical system (14) is performed under consideration of the bounds

$$u(t) = F_a\left(t\right) \in [-1 \ ; \ 1] \tag{26}$$

of the control variable. A piecewise constant control strategy $u\left(t\right) = const.$ for $t \in \Delta T \cdot [(k-1) \ ; \ k]$ is assumed, where $T = 5.0$, $\Delta T = 0.1$, and $k = 1, \ldots, 50$ are given. The performance index is defined as

$$\begin{aligned} J = & \int_0^{t_f} \left(\left(x_1\left(t\right) - 1\right)^2 + x_2\left(t\right)^2 + u\left(t\right)^2 \right) dt \\ & + 100 \Delta T \sum_{k=1}^{k_{max}} \left(u_k - u_{k-1}\right)^2 \quad . \end{aligned} \tag{27}$$

The optimization routine presented in this paper is applied to three different problems which are depicted in the Figs. 5–7. In all three cases, the dynamical system model (14) with the interval parameters $F_s \in [0.015 \ ; \ 0.050]$ and $\mu \in [0.001 \ ; \ 0.010]$ is considered.

First, the optimization is performed for the nominal system parameters $\underline{F_s}$ and $\underline{\mu}$. The resulting time response for $x_1\left(t\right)$ and $x_2\left(t\right)$ is denoted by *case (A)*. Second, the resulting control sequence is applied to the uncertain system model leading to the time responses denoted by *case (B)*. The corresponding upper bound of the performance index is given by $\overline{J} \leq 5.37$. Third, the optimization is performed directly for the uncertain parameters (*case (C)*). In this case, the resulting upper bound of the performance index is $\overline{J} \leq 7.61$. Due to the direct consideration of the uncertainties in the optimization procedure, time responses for both state variables are obtained which are overlapping completely with the desired region of admissible final states given by $[x_1\left(T\right)] = [0.9 \ ; \ 1.1]$ and $[x_2\left(T\right)] = [-0.1 \ ; \ 0.1]$.

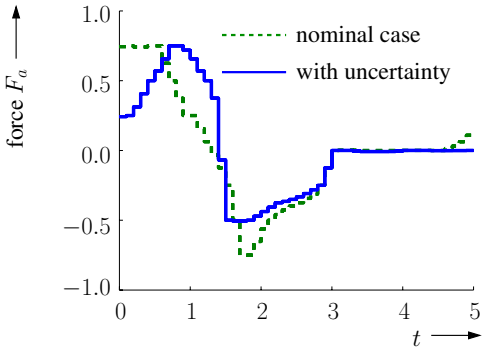


Figure 5. Optimized input variable (accelerating force F_a).

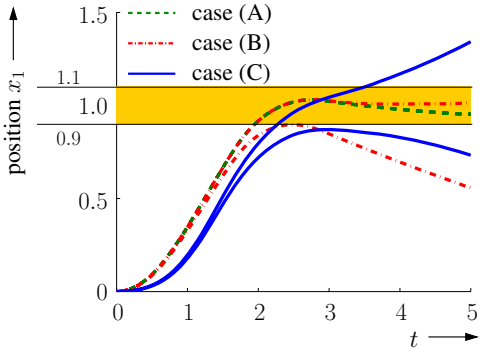


Figure 6. Resulting position x_1.

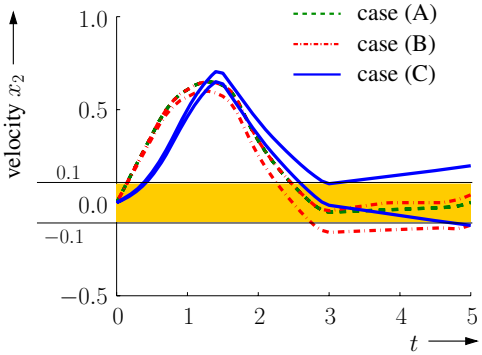


Figure 7. Resulting velocity x_2.

In the *cases (A) and (C)*, the optimization has been performed with up to 20,000 evaluations of the set of state equations for the complete time horizon using a prototypical MATLAB implementation relying on the interval arithmetic toolbox INTLAB [19, 20]. In order to reduce the computing time, the evaluation of the state equations has been vectorized such that they are evaluated for up to 20 different control sequences simultaneously.

5.3 Extension by State Controller

Since in the previously discussed cases, it is not possible to find one common control strategy for all possible system parameters such that the intervals of the final position and final velocity are completely included in $[x_1(T)] = [0.9\ ;\ 1.1]$ and $[x_2(T)] = [-0.1\ ;\ 0.1]$, an extension of the system by a closed-loop state controller is investigated. The feedback gain K of the linear state controller is determined by pole placement $\lambda_1 = \lambda_2 = -1$ using Ackermann's formula for the nominal system parameters which leads to $K = \begin{bmatrix}1 & 1.999\end{bmatrix}$. According to the block diagram in Fig. 8, the resulting accelerating force is defined as

$$u(t) = K \cdot (x_d(t) - x(t)) + u_{opt}(t) \quad , \qquad (28)$$

where u_{opt} is the result of the optimization for nominal system parameters. The reference trajectory $x_d(t)$ corresponds to the result of the optimization for nominal parameters.

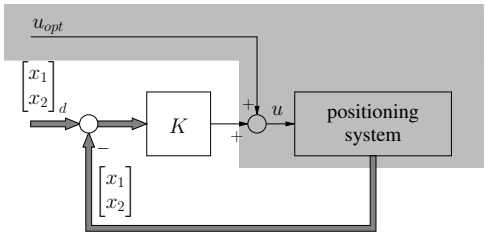


Figure 8. Block diagram for extension by linear state feedback controller.

As it is shown in the Figs. 10 and 11, the modified control law now leads to trajectories which are completely included in the region of admissible final states. This property has been proven by the extension of the Taylor series based ODE solver summarized in Subsection 5.1.

6 Conclusions and Outlook on Future Work

In this paper, an interval arithmetic optimization algorithm which is applicable to both discrete-time and

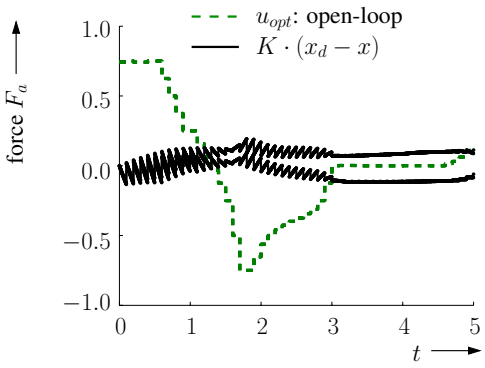


Figure 9. Accelerating force with state controller.

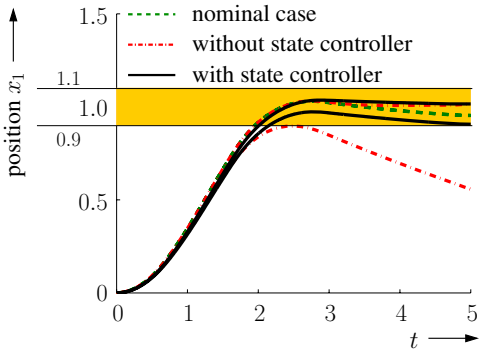


Figure 10. Uncertainty of position x_1 with state controller.

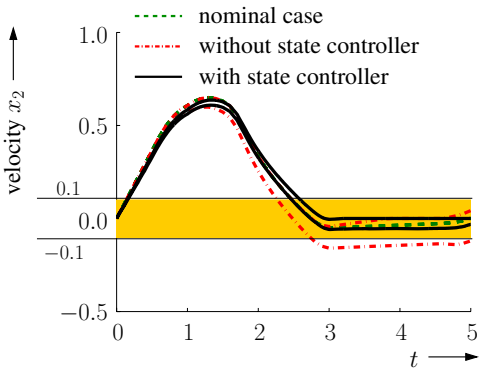


Figure 11. Uncertainty of velocity x_2 with state controller.

continuous-time dynamical systems has been presented. Furthermore, it has been applied to a simplified model of a mechanical positioning system in which state-dependent switching characteristics are included. For validated simulation of such systems, an extension of a Taylor series based validated integration algorithm is used in which the points of time are detected at which transition conditions between the discrete model states are activated or at which one of these model states becomes invalid.

In future research, the optimization routine will be extended by suitable approximations of the solution of continuous-time optimal control problems by application of Pontryagin's maximum principle to approximations of the considered nonlinear dynamical systems. The approximation techniques which will be applied are based on Carleman linearization which approximates nonlinear systems by higher-dimensional (bi-)linear models [7]. The goal of these extension is to speed up the optimization process and to reduce the computational burden in the case of high-dimensional real-world processes. Furthermore, the recently developed validated ODE solver VALENCIA-IVP will be included in the optimization routines presented in this paper in order to benefit from its main advantage which is the computation of tight state enclosures of uncertain continuous-time systems with small computational effort.

References

[1] E. Auer, A. Rauh, E. P. Hofer, and W. Luther. Validated Modeling of Mechanical Systems with SMARTMOBILE: Improvement of Performance by VALENCIA-IVP. In *Proc. of Dagstuhl Seminar 06021: Reliable Implementation of Real Number Algorithms: Theory and Practice*, Lecture Notes in Computer Science, 2006. In print.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, N. J. , 1957.

[3] R. Bellman, editor. *Mathematical Optimization Techniques*. University of California Press, Berkeley, California, 1963.

[4] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-order Taylor Models. *Reliable Computing*, 4:361–369, 1998.

[5] A. A. Feldbaum. *Optimal Control Systems*. Academic Press, New York, 1965.

[6] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter. *Applied Interval Analysis*. Springer, London, 2001.

[7] K. Kowalski and W.-H. Steeb. *Nonlinear Dynamical Systems and Carleman Linearization*. World Scientific, Singapore, 1991.

[8] G. Leitmann. *An Introduction to Optimal Control*. McGraw-Hill, New York, 1966.

[9] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, 1998.

[10] R. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[11] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 1999.

[12] N. S. Nedialkov. *The Design and Implementation of an Object-Oriented Validated ODE Solver*. Kluwer Academic Publishers, 2002.

[13] N. S. Nedialkov and M. v. Mohrenschildt. Rigorous Simulation of Hybrid Dynamic Systems with Symbolic and Interval Methods. In *Proc. of American Control Conference ACC*, pages 140–147, Anchorage, USA, 2002.

[14] L. S. Pontrjagin, V. G. Boltjanskij, R. V. Gamkrelidze, and E. F. Mišcenko. *The Mathematical Theory of Optimal Processes*. Interscience Publishers, New York, 1962.

[15] A. Rauh and E. P. Hofer. Interval Arithmetic Optimization Techniques for Uncertain Discrete-Time Systems. In E. P. Hofer and E. Reithmeier, editors, *Proc. of 13th International Workshop on Dynamics and Control*, Modeling and Control of Autonomous Decision Support Based Systems, pages 141–148, Wiesensteig, Germany, 2005. Shaker Verlag, Aachen.

[16] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Robust Controller Design for Bounded State and Control Variables and Uncertain Parameters Using Interval Methods. In *Proc. of International Conference on Control and Automation ICCA'05*, pages 777–782, Budapest, Hungary, 2005.

[17] A. Rauh, M. Kletting, H. Aschemann, and E. P. Hofer. Interval Methods for Simulation of Dynamical Systems with State-Dependent Switching Characteristics. In *Proc. of IEEE International Conference on Control Applications CCA 2006*, pages 355–360, Munich, Germany, 2006.

[18] R. Rihm. *Über Einschließungsverfahren für gewöhnliche Anfangswertprobleme und ihre Anwendung auf Differentialgleichungen mit unstetiger rechter Seite (in German)*. PhD thesis, University of Karlsruhe, Germany, 1993.

[19] S. M. Rump. IntLab (Version 5.3). `http://www.ti3.tu-harburg.de/~rump/intlab/`.

[20] S. M. Rump. INTLAB — INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluver Academic Publishers, 1999.

Deterministic Global Optimization for Dynamic Systems Using Interval Analysis

Youdong Lin and Mark A. Stadtherr
Department of Chemical and Biomolecular Engineering
University of Notre Dame, Notre Dame, IN 46556, USA
ylin@nd.edu, markst@nd.edu

Abstract

A new approach is described for the deterministic global optimization of dynamic systems, including problems in parameter estimation and optimal control. The method is based on interval analysis and Taylor models, and employs a sequential approach using a type of branch-and-reduce strategy. A key feature of the method is the use of a new validated solver for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters. This is combined with a new technique for domain reduction based on using Taylor models in an efficient constraint propagation scheme. The result is that problems can be solved to global optimality with both mathematical and computational certainty. Examples are presented to demonstrate the computational efficiency of the method.

1. Introduction

There are many applications of optimization for dynamic systems, including parameter estimation from time series data, determination of optimal operating profiles for batch and semi-batch processes, optimal start-up, shut-down, and switching of continuous system, etc. To address such problems, one approach is to discretize any control profiles that appear as decision variables. There are then basically two types of methods available: (a) the complete discretization or simultaneous approach, in which both state variables and control profiles are discretized, and (b) the control parameterization or sequential approach, in which only the control profiles are discretized. In this paper, only the sequential approach is considered. Since these problems are often nonconvex and thus may exhibit multiple local solutions, the classical techniques based on solving the necessary conditions for a local minimum may fail to determine the global optimum. This is true even for a rather simple temperature control problem with a batch reactor [10]. Therefore, there is a need to develop *global* optimization algorithms which can rigorously *guarantee* optimal performance.

The deterministic global optimization of dynamic systems has been a topic of significant recent interest. Esposito and Floudas [5, 6] used the αBB approach for addressing this problem. In this method convex underestimating functions are used in connection with a branch-and-bound framework. A theoretical guarantee of attaining an ϵ-global solution is offered as long as rigorous underestimators are used, and this requires that sufficiently large values of α be used. However, the determination of proper values of α depends on the Hessian of the function being underestimated, and, when the sequential approach is used, this matrix is not available in explicit functional form. Thus, Esposito and Floudas [5, 6] did not use rigorous values of α in their implementation of the sequential approach, and so did not obtain a theoretical guarantee of global optimality. This issue is discussed in more detail by Papamichail and Adjiman [18]. Recently, alternative approaches have been given by Chachuat and Latifi [3] and by Papamichail and Adjiman [18, 19] that provide a theoretical guarantee of ϵ-global optimality. However, this is achieved at a high computational cost. Singer and Barton [21] have recently described a branch-and-bound approach for determining an ϵ-global optimum with significantly less computational effort. In this method, convex underestimators and concave overestimators are used to construct two bounding initial value problems (IVPs), which are then solved to obtain lower and upper bounds on the trajectories of the state variables [20]. However, as implemented, the bounding IVPs are solved using standard numerical methods that do not provide guaranteed error estimates. Thus, from a computational standpoint, this approach cannot be regarded either as providing rigorously guaranteed results.

We present here a new approach for the deterministic global optimization of dynamic systems. This method is based on interval analysis and Taylor models and employs a type of sequential approach. A key feature of the method is the use of a new validated solver [8] for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parame-

ters. This is combined with a new technique for domain reduction based on using Taylor models in an efficient constraint propagation scheme. The result is that problems can be solved to global optimality with both mathematical and computational certainty.

The remainder of this paper is organized as follows. In Section 2 we present the mathematical formulation of the problem to be solved. Section 3 provides background on Taylor models. In Section 4 we review the new validated method [8] for parametric ODEs. Section 5 will then outline the algorithm for deterministic global optimization of dynamic systems. Finally, in Section 6, we present the results of some numerical experiments that demonstrate the effectiveness of the approach presented.

2. Problem statement

In this section we give the mathematical formulation of the nonlinear dynamic optimization problem to be solved. Assume the system is described by the nonlinear ODE model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$. Here $\boldsymbol{x}$ is the vector of state variables (length n) and $\boldsymbol{\theta}$ is a vector of adjustable parameters (length p), which may be a parameterization of a control profile $\theta(t)$. The model is given as an autonomous system; a non-autonomous system can easily be converted into autonomous form by treating the independent variable (t) as an additional state variable with derivative equal to 1. The objective function ϕ is expressed in terms of the adjustable parameters and the values of the states at discrete points t_μ, $\mu = 0, 1, \ldots, r$. That is, $\phi = \phi\left[\boldsymbol{x}_\mu(\boldsymbol{\theta}), \boldsymbol{\theta};\ \mu = 0, 1, \ldots, r\right]$, where $\boldsymbol{x}_\mu(\boldsymbol{\theta}) = \boldsymbol{x}(t_\mu, \boldsymbol{\theta})$. If an integral appears in the objective function, it can be eliminated by introducing an appropriate quadrature variable.

The optimization problem is then stated as

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{x}_\mu} \quad & \phi\left[\boldsymbol{x}_\mu(\boldsymbol{\theta}), \boldsymbol{\theta};\ \mu = 0, 1, \ldots, r\right] \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \\ & \boldsymbol{x}_0 = \boldsymbol{x}_0(\boldsymbol{\theta}) \\ & t \in [t_0, t_r] \\ & \boldsymbol{\theta} \in \boldsymbol{\Theta}. \end{aligned} \tag{1}$$

Here $\boldsymbol{\Theta}$ is an interval vector that provides upper and lower parameter bounds (uppercase will be used to denote interval-valued quantities, unless noted otherwise). We assume that $\boldsymbol{f}$ is $(k-1)$-times continuously differentiable with respect to the state variables $\boldsymbol{x}$, and $(q+1)$-times continuously differentiable with respect to the parameters $\boldsymbol{\theta}$. We also assume that ϕ is $(q+1)$-times continuously differentiable with respect to the parameters $\boldsymbol{\theta}$. Here k is the order of the truncation error in the interval Taylor series (ITS) method to be used in the integration procedure (to be discussed in Section 4), and q is the order of the Taylor model to be used to represent parameter dependence (to be discussed in Section 3). When a typical sequential approach is used, an ODE solver is applied to the constraints with a given set of parameter values, as determined by the optimization routine. This effectively eliminates $\boldsymbol{x}_\mu, \mu = 0, 1, \ldots, r$, and leaves a bound-constrained minimization in the adjustable parameters $\boldsymbol{\theta}$ only.

The proposed method can be extended easily to optimization problems with general state path constraints, and more general equality or inequality constraints on parameters. This is done by adapting the constraint propagation procedure discussed in Section 5.1 to handle the additional constraints.

3. Taylor models

Makino and Berz have described a remainder differential algebra (RDA) approach that uses Taylor models for bounding function ranges and control of the dependency problem of interval arithmetic [11]. In this method, a function is represented using a model consisting of a Taylor polynomial and an interval remainder bound.

One way of forming a Taylor model of a function is by using a truncated Taylor series. Consider a function $f : \boldsymbol{x} \in \boldsymbol{X} \subset \mathbb{R}^m \to \mathbb{R}$ that is $(q+1)$ times partially differentiable on $\boldsymbol{X}$ and let $\boldsymbol{x}_0 \in \boldsymbol{X}$. The Taylor theorem states that for each $\boldsymbol{x} \in \boldsymbol{X}$, there exists a $\zeta \in \mathbb{R}$ with $0 < \zeta < 1$ such that

$$\begin{aligned} f(\boldsymbol{x}) \ = \ & \sum_{i=0}^{q} \frac{1}{i!} \left[(\boldsymbol{x} - \boldsymbol{x}_0) \cdot \bigtriangledown\right]^i f(\boldsymbol{x}_0) \\ + \ & \frac{1}{(q+1)!} \left[(\boldsymbol{x} - \boldsymbol{x}_0) \cdot \bigtriangledown\right]^{q+1} f\left[\boldsymbol{x}_0 + (\boldsymbol{x} - \boldsymbol{x}_0)\zeta\right], \end{aligned} \tag{2}$$

where the partial differential operator $[\boldsymbol{g} \cdot \bigtriangledown]^k$ is

$$[\boldsymbol{g} \cdot \bigtriangledown]^k = \sum_{\substack{j_1 + \cdots + j_m = k \\ 0 \le j_1, \cdots, j_m \le k}} \frac{k!}{j_1! \cdots j_m!} g_1^{j_1} \cdots g_m^{j_m} \frac{\partial^k}{\partial x_1^{j_1} \cdots \partial x_m^{j_m}}. \tag{3}$$

The last (remainder) term in Eq. (2) can be quantitatively bounded over $0 < \zeta < 1$ and $\boldsymbol{x} \in \boldsymbol{X}$ using interval arithmetic or other methods to obtain an interval remainder bound R_f. The summation in Eq. (2) is a q-th order polynomial (truncated Taylor series) in $(\boldsymbol{x} - \boldsymbol{x}_0)$ which we denote by $p_f(\boldsymbol{x} - \boldsymbol{x}_0)$. A q-th order Taylor model T_f for $f(\boldsymbol{x})$ then consists of the polynomial p_f and the interval remainder bound R_f and is denoted by $T_f = (p_f, R_f)$. Note that $f \in T_f$ for $\boldsymbol{x} \in \boldsymbol{X}$ and thus T_f encloses the range of f over $\boldsymbol{X}$.

In practice, it is more useful to compute Taylor models of functions by performing Taylor model operations. Arithmetic operations with Taylor models can be done using the RDA operations described by Makino and Berz [11, 12],

which include addition, multiplication, reciprocal, and intrinsic functions. Therefore, it is possible to compute a Taylor model for any function representable in a computer environment by simple operator overloading through RDA operations. In performing RDA operations, only the coefficients of p_f are stored and operated on. However, rounding errors are bounded and added to R_f. It has been shown that, compared to other rigorous bounding methods, the Taylor model can be used to obtain sharper bounds for modest to complicated functional dependencies [11, 17].

An interval bound on a Taylor model $T = (p, R)$ over $\boldsymbol{X}$ is denoted by $B(T)$, and is found by determining an interval bound $B(p)$ on the polynomial part p and then adding the remainder bound; that is $B(T) = B(p) + R$. The range bounding of the polynomials $B(p) = P(\boldsymbol{X} - \boldsymbol{x}_0)$ is an important issue, which directly affects the performance of Taylor model methods. Unfortunately, exact range bounding of an interval polynomial is NP hard, and direct evaluation using interval arithmetic is very inefficient, often yielding only loose bounds. Thus, various bounding schemes [13, 17] have been used, mostly focused on exact bounding of the dominant parts of P, i.e., the first- and second-order terms. However, exact bounding of a general interval quadratic is also computationally expensive (in the worst case, exponential in the number of variables m). Thus, we have adopted here a very simple compromise approach, in which only the first-order and the diagonal second-order terms are considered for exact bounding, and other terms are evaluated directly. That is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0}\right)^2 + b_i (X_i - x_{i0}) \right] + Q, \quad (4)$$

where Q is the interval bound of all other terms, and is obtained by direct evaluation with interval arithmetic. In Eq. (4), since X_i occurs twice, there exists a dependency problem. For $|a_i| \geq \omega$, where ω is a small positive number, we can rearrange Eq. (4) such that each X_i occurs only once; that is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right] + Q. \quad (5)$$

In this way, the dependence problem in bounding the interval polynomial is alleviated so that a sharper bound can be obtained. If $|a_i| < \omega$, direct evaluation will be used instead.

4. Validating solver for parametric ODEs

When a traditional sequential approach is applied to the optimization of nonlinear dynamic systems, the objective function ϕ is evaluated, for a given value of $\boldsymbol{\theta}$, by applying an ODE solver to the constraints to eliminate the state variables $\boldsymbol{x}$. In the global optimization algorithm described here, we will use a sequential approach based on interval analysis. This approach requires the evaluation of bounds on ϕ, given some parameter interval $\boldsymbol{\Theta}$. Thus, we need an ODE solver that can compute bounds on $\boldsymbol{x}_\mu, \mu = 0, 1, \ldots, r$, for the case in which the parameters are interval valued. Interval methods (these are validated methods or verified methods) for ODEs [14], provide a natural approach for computing the desired enclosure of the state variables at $t_\mu, \mu = 0, 1, \ldots, r$. An excellent review of interval methods for IVPs has been given by Nedialkov et al. [15]. Much work has been done for the case in which the initial values are given by intervals, and there are several available software packages that deal with this case. However, relatively little work has been done on the case in which parameters are also given by intervals. In our method for deterministic global optimization of dynamic systems, we will use a new validated solver for parametric ODEs [8], called VSPODE (Validating Solver for Parametric ODEs), which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued initial states and parameters. In this section, we review the key ideas behind the new method used in VSPODE, and outline the procedures used. Additional details are given by Lin and Stadtherr [8].

Consider the parametric ODE system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \quad \boldsymbol{x}_0 \in \boldsymbol{X}_0, \quad \boldsymbol{\theta} \in \boldsymbol{\Theta}, \quad (6)$$

where $t \in [t_0, t_r]$ for some $t_r > t_0$. The interval vectors $\boldsymbol{X}_0$ and $\boldsymbol{\Theta}$ represent enclosures of initial values and parameters, respectively. It is desired to determine a validated enclosure of all possible solutions to this initial value problem. We denote by $\boldsymbol{x}(t; t_j, \boldsymbol{X}_j, \boldsymbol{\Theta})$ the set of solutions $\boldsymbol{x}(t; t_j, \boldsymbol{X}_j, \boldsymbol{\Theta}) = \{\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \mid \boldsymbol{x}_j \in \boldsymbol{X}_j, \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$, where $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta})$ denotes a solution of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$ for the initial condition $\boldsymbol{x} = \boldsymbol{x}_j$ at $t = t_j$. We will outline a method for determining enclosures $\boldsymbol{X}_j$ of the state variables at each time step $j = 1, \ldots, r$, such that $\boldsymbol{x}(t_j; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{X}_j$.

Assume that at t_j we have an enclosure $\boldsymbol{X}_j$ of $\boldsymbol{x}(t_j; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta})$, and that we want to carry out an integration step to compute the next enclosure $\boldsymbol{X}_{j+1}$. Then, in the first phase of the method, the goal is to find a step size $h_j = t_{j+1} - t_j > 0$ and an a priori enclosure (coarse enclosure) $\widetilde{\boldsymbol{X}}_j$ of the solution such that a unique solution $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \in \widetilde{\boldsymbol{X}}_j$ is guaranteed to exist for all $t \in [t_j, t_{j+1}]$, all $\boldsymbol{x}_j \in \boldsymbol{X}_j$, and all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. We apply the traditional interval method, with high order enclosure, to the parametric ODEs by using an interval Taylor series (ITS) with respect to time. That is, we determine h_j and

$\boldsymbol{X}_j$ such that for $\boldsymbol{X}_j \subseteq \widetilde{\boldsymbol{X}}_j^0$,

$$\widetilde{\boldsymbol{X}}_j = \sum_{i=0}^{k-1} [0,h_j]^i \boldsymbol{F}^{[i]}(\boldsymbol{X}_j, \boldsymbol{\Theta}) + [0,h_j]^k \boldsymbol{F}^{[k]}(\widetilde{\boldsymbol{X}}_j^0, \boldsymbol{\Theta}) \subseteq \widetilde{\boldsymbol{X}}_j^0. \tag{7}$$

Here $\widetilde{\boldsymbol{X}}_j^0$ is an initial estimate of $\widetilde{\boldsymbol{X}}_j$, k denotes the order of the Taylor expansion, and the coefficients $\boldsymbol{F}^{[i]}$ are interval extensions of the Taylor coefficients $\boldsymbol{f}^{[i]}$ of $\boldsymbol{x}(t)$ with respect to time. Satisfaction of Eq. (7) demonstrates [4] that there exists a unique solution $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \in \widetilde{\boldsymbol{X}}_j$ for all $t \in [t_j, t_{j+1}]$, all $\boldsymbol{x}_j \in \boldsymbol{X}_j$, and all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

In the second phase of the method, we compute a tighter enclosure $\boldsymbol{X}_{j+1} \subseteq \widetilde{\boldsymbol{X}}_j$, such that $\boldsymbol{x}(t_{j+1}; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{X}_{j+1}$. This will be done by using an ITS approach to compute a Taylor model $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ of $\boldsymbol{x}_{j+1}$ in terms of the parameter vector $\boldsymbol{\theta}$ and initial state vector $\boldsymbol{x}_0$, and then obtaining the enclosure $\boldsymbol{X}_{j+1} = B(\boldsymbol{T}_{\boldsymbol{x}_{j+1}})$ by bounding $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ over $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\boldsymbol{x}_0 \in \boldsymbol{X}_0$. To determine enclosures of the interval Taylor series coefficients $\boldsymbol{f}^{[i]}(\boldsymbol{x}_j, \boldsymbol{\theta})$ a novel approach combining RDA operations with the mean value theorem is used to obtain the Taylor models $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$. Now using an interval Taylor series for $\boldsymbol{x}_{j+1}$ with coefficients given by $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$, one can obtain a result for $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ in terms of the parameters and initial states. In order to address the wrapping effect [14], results are propagated from one time step to the next using a new type of Taylor model, in which the remainder bound is not an interval, but a parallelepiped. That is, the remainder bound is a set of the form $\mathcal{P} = \{Av \mid v \in V\}$, where $A \in \mathbb{R}^{n \times n}$ is a real and regular matrix. If A is orthogonal, as from a QR-factorization, then $\mathcal{P}$ can be interpreted as a rotated n-dimensional rectangle. Complete details of the computation of $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ are given by Lin and Stadtherr [8].

The approach outlined above, as implemented in VSPODE, has been tested by Lin and Stadtherr [8], who compared its performance with results obtained using the popular VNODE package [16]. For the test problems used, VSPODE provided tighter enclosures on the state variables than VNODE, and required significantly less computation time.

5. Deterministic global optimization method

In this section, we summarize a new method for the deterministic global optimization of dynamic systems. As noted above, when a sequential approach is used, the state variables are effectively eliminated using the ODE constraints, in this case by employing VSPODE, leaving a bound-constrained minimization of $\phi(\boldsymbol{\theta})$ with respect to the adjustable parameters (decision variables) $\boldsymbol{\theta}$. The new approach can be thought of as a type of branch-and-bound method, with a constraint propagation procedure used for domain reduction. Therefore, it can also be viewed as a branch-and-reduce algorithm. The basic idea is that only those parts of the decision variable space $\boldsymbol{\Theta}$ that satisfy the constraint $c(\boldsymbol{\theta}) = \phi(\boldsymbol{\theta}) - \widehat{\phi} \leq 0$, where $\widehat{\phi}$ is a known upper bound on the global minimum, need to be retained. We now describe a constraint propagation procedure, based on the use of Taylor models, that exploits this constraint information for domain reduction.

5.1. Constraint propagation on Taylor models

Partial information expressed by a constraint can be used to eliminate incompatible values from the domain of its variables. This domain reduction can then be propagated to all constraints on that variable, where it may be used to further reduce the domains of other variables. This process is known as constraint propagation. In this subsection, we describe a constraint propagation procedure using Taylor models. In this discussion, an underline is used to indicate the lower (left) endpoint of an interval, and an overline is used to indicate the upper (right) endpoint.

Let T_c be the Taylor model of the function $c(\boldsymbol{x})$ over the interval $\boldsymbol{x} \in \boldsymbol{X}$, and say the constraint $c(\boldsymbol{x}) \leq 0$ needs to be satisfied. In the constraint propagation procedure (CPP) described here, $B(T_c)$ is determined and then there are three possible outcomes: 1. If $\underline{B(T_c)} > 0$, then no $\boldsymbol{x} \in \boldsymbol{X}$ will ever satisfy the constraint; thus, the CPP can be stopped and $\boldsymbol{X}$ discarded. 2. If $\overline{B(T_c)} \leq 0$, then every $\boldsymbol{x} \in \boldsymbol{X}$ will always satisfy the constraint; thus $\boldsymbol{X}$ cannot be reduced and the CPP can be stopped. 3. If neither of previous two cases occur, then part of the interval $\boldsymbol{X}$ may be eliminated; thus the CPP continues, using an approach based on the range bounding strategy for Taylor models described above.

For some component i of $\boldsymbol{x}$, let a_i and b_i be the polynomial coefficients of the terms $(x_i - x_{i0})^2$ and $(x_i - x_{i0})$ of T_c, respectively. Note that, $x_{i0} \in X_i$ and is usually the midpoint $x_{i0} = m(X_i)$; the value of x_{i0} will not change during the CPP. For $|a_i| \geq \omega$, the bounds on T_c can be expressed using Eq. (5) as

$$B(T_c) = a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} + S_i, \tag{8}$$

where S_i is an interval bound on all the other terms of T_c. We can reduce the computational effort to obtain S_i by recognizing that this quantity is just $B(T_c)$ less the i-th term in the summation of Eq. (5), and $B(T_c)$ was already computed earlier in the CPP. Therefore, for each i, S_i is determined by the interval cancellation (dependent subtraction) operation (denoted by $\ominus$) using

$$S_i = B(T_c) \ominus \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right]. \tag{9}$$

Now define the intervals $U_i = X_i - x_{i0} + \frac{b_i}{2a_i}$ and $V_i = \frac{b_i^2}{4a_i} - S_i$, so that $B(T_c) = a_i U_i^2 - V_i$. The goal is to identify and retain only the part of X_i that contains values of x_i for which it is possible to satisfy $c(\boldsymbol{x}) \leq 0$. In other words, the part of X_i that is going to be eliminated is guaranteed not to satisfy the constraint $c(\boldsymbol{x}) \leq 0$. Since $B(T_c) = a_i U_i^2 - V_i$ bounds the range of $c(\boldsymbol{x})$ for $\boldsymbol{x} \in \boldsymbol{X}$, the part of X_i in which it is possible to satisfy $c(\boldsymbol{x}) \leq 0$ can be bounded by finding X_i such that all elements of $a_i U_i^2$ are less than or equal to at least one element of V_i. That is, we require that

$$\overline{a_i U_i^2} \leq \overline{V_i}. \tag{10}$$

Then, the set U_i that satisfies Eq. (10) can be determined to be

$$U_i = \begin{cases} \varnothing & \text{if } a_i > 0 \text{ and } \overline{V_i} < 0 \\ \left[-\sqrt{\frac{\overline{V_i}}{a_i}}, \sqrt{\frac{\overline{V_i}}{a_i}}\right] & \text{if } a_i > 0 \text{ and } \overline{V_i} \geq 0 \\ [-\infty, \infty] & \text{if } a_i < 0 \text{ and } \overline{V_i} \geq 0 \\ \left[-\infty, -\sqrt{\frac{\overline{V_i}}{a_i}}\right] \cup \left[\sqrt{\frac{\overline{V_i}}{a_i}}, \infty\right] & \text{if } a_i < 0 \text{ and } \overline{V_i} < 0. \end{cases} \tag{11}$$

The part of X_i to be retained is then $X_i = X_i \cap \left(U_i + x_{i0} - \frac{b_i}{2a_i}\right)$. If $|a_i| < \omega$, then Eq. (5) should not be used, but if $|b_i| \geq \omega$, then Eq. (4) can, and a procedure similar to that used above can be developed to detemine the part of X_i to be retained [7]. If both $|a_i|$ and $|b_i|$ are less than ω, then no CPP will be applied on X_i.

The overall CPP is implemented by beginning with $i = 1$ and proceeding component by component. If, for any i, the result $X_i = \varnothing$ is obtained, then no $\boldsymbol{x} \in \boldsymbol{X}$ can satisfy the constraint; thus, $\boldsymbol{X}$ can be discarded and the CPP stopped. Otherwise the CPP proceeds until all components of $\boldsymbol{X}$ have been updated. Note that, in principle, each time an improved (smaller) X_i is found, it could be used in computing S_i for subsequent components of $\boldsymbol{X}$. However, this requires recomputing the bound $B(T_c)$, which, for the function $c(\boldsymbol{x})$ that is of interest here, is expensive. Thus, the CPP for each component is done using the bounds $B(T_c)$ computed from the original $\boldsymbol{X}$. If, after each component is processed, $\boldsymbol{X}$ has been sufficiently reduced (by more than $\omega_1 = 10\%$ by volume), then a new bound $B(T_c)$ is obtained, now over the smaller $\boldsymbol{X}$, and a new CPP is started. Otherwise, the CPP terminates.

5.2. Global optimization algorithm

As with any type of procedure incorporating branch-and-bound, an important issue is how to initialize $\widehat{\phi}$, the upper bound on the global minimum. There are many ways in which this can be done, and clearly, it is desirable to find a $\widehat{\phi}$ that is as small as possible (i.e., the tightest possible upper bound). To initialize $\widehat{\phi}$, we run p^2 local minimizations (p is the number of adjustable parameters) using a local optimization routine from randomly chosen starting points, and then choose the smallest value of ϕ found to be the initial $\widehat{\phi}$. For this purpose, we use the bound-constrained quasi-Newton method L-BFGS-B [2] as the local optimization routine, and DDASSL [1] as the integration routine. Additional initialization steps are to set either a relative convergence tolerance ϵ^{rel} or an absolute convergence tolerance ϵ^{abs}, and to initialize a work list $\mathcal{L}$. The work list (stack) $\mathcal{L}$ will contain a sequence of subintervals (boxes) that need to be tested and initially $\mathcal{L} = \{\boldsymbol{\Theta}\}$, the entire parameter (decision variable) space.

The core steps in the iterative process involve the testing of boxes in the work list. This is an objective range test combined with domain reduction done using the CPP described above. Beginning with $k = 0$, at the k-th iteration a box is removed from the front of $\mathcal{L}$ and is designated as the current subinterval $\boldsymbol{\Theta}^{(k)}$. The Taylor model T_{ϕ_k} of the objective function ϕ over $\boldsymbol{\Theta}^{(k)}$ is computed. To do this, Taylor models of $\boldsymbol{x}_\mu$, the state variables at times $t_\mu, \mu = 1, \ldots, r$, in terms of $\boldsymbol{\theta}$ are determined using VSPODE, as described in Section 4. Note that T_{ϕ_k} then consists of a q-th order polynomial in the decision variables $\boldsymbol{\theta}$, plus a remainder bound. The part of $\boldsymbol{\Theta}^{(k)}$ that can contain the global minimum must satisfy the constraint $c(\boldsymbol{\theta}) = \phi(\boldsymbol{\theta}) - \widehat{\phi} \leq 0$ Thus the constraint propagation procedure (CPP) described in Section 5.1 is now applied using this constraint. Recall that there are three possible outcomes in the CPP:

1. Testing for the first possible outcome, $\underline{B(T_c)} > 0$, amounts to checking if the lower bound of T_{ϕ_k}, $\underline{\overline{B(T_{\phi_k})}}$, is greater than $\widehat{\phi}$. If so, then $\boldsymbol{\Theta}^{(k)}$ can be discarded because it cannot contain the global minimum and need not be further tested.

2. Testing for the second possible outcome, $\overline{B(T_c)} \leq 0$, amounts to checking if the upper bound of T_{ϕ_k}, $\overline{B(T_{\phi_k})}$, is less than $\widehat{\phi}$. If so, then all points in $\boldsymbol{\Theta}^{(k)}$ satisfy the constraint and the CPP can be stopped since no reduction in $\boldsymbol{\Theta}^{(k)}$ can be achieved. This also indicates, with certainty, that there is a point in $\boldsymbol{\Theta}^{(k)}$ that can be used to update $\widehat{\phi}$. Thus, if $\overline{B(T_{\phi_k})} < \widehat{\phi}$, a local optimization routine, starting at some point in $\boldsymbol{\Theta}^{(k)}$, is used to find a local minimum, which then provides an updated (smaller) $\widehat{\phi}$, that is, a better upper bound on the global minimum. In our implementation, the midpoint of $\boldsymbol{\Theta}^{(k)}$ is used as the starting point for the local optimization. A new CPP is then started on $\boldsymbol{\Theta}^{(k)}$ using the updated value of $\widehat{\phi}$.

3. If neither of the previous two outcomes occurs, then the full CPP described in Section 5.1 is applied to reduce $\boldsymbol{\Theta}^{(k)}$. Note that if $\boldsymbol{\Theta}^{(k)}$ is sufficiently reduced (by more than $\omega_1 = 10\%$ by volume) in comparison to its volume at the beginning of CPP, then new bounds $B(T_{\phi_k})$ are ob-

tained, now over the smaller $\mathbf{\Theta}^{(k)}$, and a new CPP is started.

After the CPP terminates, a convergence test is performed. If $(\widehat{\phi} - \underline{B(T_{\phi_k})})/|\widehat{\phi}| \leq \epsilon^{\text{rel}}$, or $(\widehat{\phi} - \underline{B(T_{\phi_k})}) \leq \epsilon^{\text{abs}}$, then $\mathbf{\Theta}^{(k)}$ need not be further tested and can be discarded. Otherwise, we will check to what extent $\mathbf{\Theta}^{(k)}$ has been reduced compared to its volume at the beginning of the objective range test. If the subinterval is reduced by more than $\omega_2 = 70\%$ by volume, it will be added to the beginning of the sequence $\mathcal{L}$ of boxes to be tested. Otherwise, it will be bisected, and the resulting two subintervals added to the beginning of $\mathcal{L}$. Various strategies can be used to select the component to be bisected. For the problems solved here, the component with the largest relative width was selected for bisection. The volume reduction targets ω_1 and ω_2 can be adjusted as needed to tune the algorithm; the default values given above were used in the computational studies described below.

At the end of this testing process, k is incremented, a box is removed from the front of $\mathcal{L}$, and the testing process is begun again. At termination, $\mathcal{L}$ will become empty, and $\widehat{\phi}$ is the ϵ-global minimum.

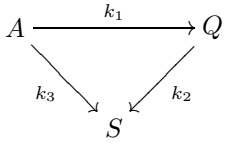
The method described above is an ϵ-global algorithm. It is also possible to incorporate interval-Newton steps in the method, and to thus make it an exact ($\epsilon = 0$) algorithm. This requires the application of VSPODE on the first- and second-order sensitivity equations. An exact algorithm using interval-Newton steps has been implemented by Lin and Stadtherr [7] for the special case of parameter estimation problems, such as shown in Section 6.1. However, this has not yet been fully implemented for more general cases, such as those in Section 6.2.

6. Computational studies

In this section, two example problems are presented to demonstrate the computational performance of the approach described above. Both example problems were solved on an Intel Pentium 4 3.2 GHz machine running Red Hat Linux. The VSPODE package [8], with a $k = 17$ order interval Taylor series, $q = 3$ order Taylor model, and QR approach for wrapping, was used to integrate the dynamic systems in each problem. Using a smaller value of k will result in the need for smaller step sizes in the integration and so will tend to increase computation time. Using a larger value of q will result in somewhat tighter bounds on the states, though at the expense of additional complexity in the Taylor model computations. The algorithm was implemented in C++.

6.1. Catalytic cracking of gas oil

This problem involves parameter estimation in a model representing the catalytic cracking of gas oil (A) to gasoline (Q) and other side products (S), as described by Tjoa and Biegler [22] and also studied by several others [3, 6, 19, 21]. The reaction is



Only the concentrations of A and Q were measured. This reaction scheme involves nonlinear reaction kinetics. A least squares objective was used for parameter estimation, resulting in the optimization problem

$$\begin{aligned}
\min_{\boldsymbol{\theta}} \phi \quad &= \quad \sum_{\mu=1}^{20} \sum_{i=1}^{2} \left(\widehat{x}_{\mu,i} - x_{\mu,i}\right)^2 \\
\text{s.t.} \quad \dot{x}_1 \quad &= -(\theta_1 + \theta_3) x_1^2 \\
\dot{x}_2 \quad &= \theta_1 x_1^2 - \theta_2 x_2 \\
t \quad &\in [0, 0.95] \\
\boldsymbol{x}_\mu \quad &= \boldsymbol{x}(t_\mu) \\
\boldsymbol{x}_0 \quad &= (1, 0)^{\mathrm{T}} \\
\boldsymbol{\theta} \quad &\in [0, 20] \times [0, 20] \times [0, 20],
\end{aligned}$$

where $\widehat{\boldsymbol{x}}_\mu$ is given experimental data. Here the state vector, $\boldsymbol{x}$, is defined as the concentration vector $(A, Q)^{\mathrm{T}}$ and the parameter vector, $\boldsymbol{\theta}$, is defined as $(k_1, k_2, k_3)^{\mathrm{T}}$. The measurement data $\widehat{\boldsymbol{x}}_\mu$ was generated using values of the parameters, $\boldsymbol{\theta} = (12, 8, 2)^{\mathrm{T}}$, with a small amount of random error added, and can be found in Esposito and Floudas [6]. We solved this problem with the ϵ-global optimization procedure discussed above, and also with the exact ($\epsilon = 0$) algorithm [7] that incorporates interval-Newton steps.

A summary of computational results, with comparisons to other methods, is given in Table 1. In all cases, the global minimum found was 2.6557×10^{-3} with the parameter values of $\boldsymbol{\theta} = (12.2139, 7.9798, 2.2217)^{\mathrm{T}}$, which is consistent with the result of Esposito and Floudas [6]. For the ϵ-global algorithm, 14.3 seconds were required to solve the problem. For the exact global algorithm using interval-Newton, 11.5 seconds were required. For this problem, the exact algorithm required less computation than the ϵ-global algorithm. However, this may or may not be the case on other problems [7].

Papamichail and Adjiman [19] solved this problem to ϵ-global optimality in 35478 seconds (Sun UltraSPARC-II 360 MHz; Matlab), and Chachuat and Latifi [3] obtained an ϵ-global solution in 10400 seconds (unspecified machine; prototype implementation). Singer and Barton [21] solved this problem to ϵ-global optimality for a series of absolute tolerances, so their results are not directly comparable. However, the computational cost of their method on

Table 1. Results for the catalytic cracking of gas oil problem.

Method	CPU time (s) Reported	Adjusted*
This work (exact global optimum: $\epsilon = 0$) (Intel P4 3.2GHz)	11.5	11.5
This work ($\epsilon^{\mathrm{rel}} = 10^{-3}$) (Intel P4 3.2GHz)	14.3	14.3
Papamichail and Adjiman (2002) ($\epsilon^{\mathrm{rel}} = 10^{-3}$) (SUN UltraSPARC-II 360MHz)	35478	4541
Chachuat and Latifi (2003) ($\epsilon^{\mathrm{rel}} = 10^{-3}$) (Machine not reported)	10400	–
Singer and Barton (2006) ($\epsilon^{\mathrm{abs}} = 10^{-3}$) (AMD Athlon 2000XP+ 1.667GHz)	5.78	2.89

*Adjusted = Approximate CPU time after adjustment for machine used (based on SPEC benchmark)

this problem appears to be quite low. These other methods all provide for ϵ-convergence only. The use of physical state bounds, known independently of the statement of the optimization problem, is an important feature of the method used by Singer and Barton [21], and this contributes to its efficiency on some problems. We have not made use of any physical state bounds in the example problems presented here.

6.2. Oil shale pyrolysis problem

This example is a fixed final time formulation of the oil shale pyrolysis problem originally formulated by Luus [9] and also considered by Esposito and Floudas [5] and Singer and Barton [21]. The problem formulation is:

$$\begin{aligned} \min_{\theta(t)} \quad & \phi && = -x_2(t_f) \\ \text{s.t.} \quad & \dot{x}_1 && = -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \\ & \dot{x}_2 && = k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \\ & k_i && = a_i \exp\left(\frac{-b_i/R}{\theta}\right), i = 1, \ldots, 5 \\ & \boldsymbol{x}_0 && = (1, 0)^{\mathrm{T}} \\ & t && \in [t_0, t_f] = [0, 10] \\ & \theta && \in [698.15, 748.15]. \end{aligned} \tag{12}$$

The values for a_i and b_i/R are defined in [5].

In Problem (12), a reciprocal operation on the control variable is required to calculate the k_i, which imposes a significant overhead when the related Taylor model computations are done. Thus, for the control variable we use the simple transformation $\bar{\theta} = 698.15/\theta$. The transformed problem then becomes:

$$\begin{aligned} \min_{\bar{\theta}(t)} \quad & \phi && = -x_2(t_f) \\ \text{s.t.} \quad & \dot{x}_1 && = -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \\ & \dot{x}_2 && = k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \\ & k_i && = a_i \exp\left(-\bar{\theta} b_i/R\right), i = 1, \cdots, 5 \\ & \boldsymbol{x}_0 && = (1, 0)^{\mathrm{T}} \\ & t && \in [t_0, t_f] = [0, 10] \\ & \bar{\theta} && \in [698.15/748.15,\ 1]. \end{aligned} \tag{13}$$

The control $\bar{\theta}(t)$ was parameterized as a piecewise constant profile with a specified number of equal time intervals. Four problems were considered, corresponding to one, two, three and four time intervals in the parameterization. Each problem was solved to an absolute tolerance of $\epsilon^{\mathrm{abs}} = 10^{-3}$. The results are presented in Table 2.

Singer and Barton [21] solved the one- and two-interval cases with $\epsilon^{\mathrm{abs}} = 10^{-3}$ using two different implementations (with and without branch-and-bound heuristics). Best results in terms of efficiency were achieved using the heuristics, with CPU times of 26.2 and 1597.3 seconds (1.667 GHz AMD Athlon XP2000+) on the one- and two-interval problems, respectively. This compares to CPU times of 3.2 and 26.8 seconds (3.2 GHz Intel Pentium 4) for the method given here. Even after accounting for the roughly factor of two difference in the speeds of the machines used, the method described here appears to be significantly more efficient, by well over one order of magnitude in the two-interval case. The three- and four-interval problems were solved here in 251.6 and 2443.5 CPU seconds, respectively, and apparently have not been solved previously using a rigorously guaranteed method. The worst-case exponential complexity seen in these results reflects the fact that global optimization for nonlinear problems is in general NP-hard.

Table 2. Results for the oil shale pyrolysis problem.

# of time intervals	ϕ^*	$\bar{\theta}^*$	CPU (s)	Iterations
1	-0.3479	(0.984)	3.2	21
2	-0.3510	(0.970, 1.000)	26.8	178
3	-0.3517	(1.000, 0.963, 1.000)	251.6	1 531
4	-0.3523	(1.000, 0.9545, 1.000, 1.000)	2443.5	12 874

7. Concluding remarks

We have presented here a new approach for the deterministic global optimization of dynamic systems, including parameter estimation and optimal control problems. This method is based on interval analysis and Taylor models and employs a type of sequential approach. A key feature of the method is the use of a new validated solver [8] for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters. This is combined with a new technique for domain reduction based on using Taylor models in an efficient constraint propagation scheme. The result is that problems can be solved to global optimality with both mathematical and computational certainty. On parameter estimation problems, an exact ($\epsilon = 0$) algorithm, using interval-Newton steps, can be applied at a cost comparable to, and perhaps less than, the ϵ-global algorithm. The new approach can provide significant improvements in computational efficiency, potentially well over one order of magnitude, relative to other recently decribed methods.

Acknowledgments

This work was supported in part by the State of Indiana 21st Century Research and Technology Fund under Grant #909010455, and by the Department of Energy under Grant DE-FG02-05CH11294.

References

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, 1996.

[2] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208, 1995.

[3] B. Chachuat and M. A. Latifi. A new approach in deterministic global optimisation of problems with ordinary differential equations. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, pages 83–108, Dordrecht, The Netherlands, 2004. Kluwer Academic Publishers.

[4] G. F. Corliss and R. Rihm. Validating an a priori enclosure using high-order Taylor series. In G. Alefeld and A. Frommer, editors, *Scientific Computing: Computer Arithmetic, and Validated Numerics*, pages 228–238. Akademie Verlag, Berlin, 1996.

[5] W. R. Esposito and C. A. Floudas. Deterministic global optimization in nonlinear optimal control problems. *J. Global Optim.*, 17:97–126, 2000.

[6] W. R. Esposito and C. A. Floudas. Global optimization for the parameter estimation of differential-algebraic systems. *Ind. Eng. Chem. Res.*, 39:1291–1310, 2000.

[7] Y. Lin and M. A. Stadtherr. Determinstic global optimization for parameter estimation of dynamic systems. *Ind. Eng. Chem. Res.*, 45:8438–8448, 2006.

[8] Y. Lin and M. A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Appl. Numer. Math.*, in press, 2007.

[9] R. Luus. Optimal control by dynamic programming using systematic reduction in grid size. *Int. J. Control*, 51:995–1013, 1990.

[10] R. Luus and D. E. Cormack. Multiplicity of solutions resulting from the use of variational methods in optimal control problems. *Can. J. Chem. Eng.*, 50:309–311, 1972.

[11] K. Makino and M. Berz. Efficient control of the dependency problem based on Taylor model methods. *Reliab. Comput.*, 5:3–12, 1999.

[12] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *Int. J. Pure Appl. Math.*, 4:379–456, 2003.

[13] K. Makino and M. Berz. Verified global optimization with Taylor model-based range bounders. *Transactions on Computers*, 11:1611–1618, 2005.

[14] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[15] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Appl. Math. Comput.*, 105:21–68, 1999.

[16] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliab. Comput.*, 7:449–465, 2001.

[17] A. Neumaier. Taylor forms – Use and limits. *Reliab. Comput.*, 9:43–79, 2003.

[18] I. Papamichail and C. S. Adjiman. A rigorous global optimization algorithm for problems with ordinary differential equations. *J. Global Optim.*, 24:1–33, 2002.

[19] I. Papamichail and C. S. Adjiman. Global optimization of dynamic systems. *Comput. Chem. Eng.*, 28:403–415, 2004.

[20] A. B. Singer and P. I. Barton. Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM J. Sci. Comput.*, 27:2167–2182, 2006.

[21] A. B. Singer and P. I. Barton. Global optimization with nonlinear ordinary differential equations. *J. Global Optim.*, 34:159–190, 2006.

[22] T. B. Tjoa and L. T. Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Ind. Eng. Chem. Res.*, 30:376–385, 1991.

Computing the Jordan canonical form in finite precision arithmetic

Toshio Suzuki
University of Yamanashi
Faculty of Education & Human Sciences
Kofu 400-8510, Japan
suzuki@yamanashi.ac.jp

Tomohiro Suzuki
University of Yamanashi
Information Processing Center
Kofu 400-8511, Japan
stomo@yamanashi.ac.jp

Abstract

The authors propose a criterion how to decide a cluster of eigenvalues to be a multiple eigenvalue or nearly multiple eigenvalues in finite precision arithmetic. If the matrix has a multiple eigenvalue, the eigenvector and the generalized ones are computed by their method, and therefore the Jordan canonical form can be derived. Results of numerical experiments for several kinds of matrices are shown.

1. Introduction

In [6] we showed that if the eigenvalue of a matrix is defective and derogatory, our method works effectively to compute the eigenvalue, the eigenvectors and generalized eigenvectors. But there exists a problem for general matrices to decide numerically whether a cluster of eigenvalues is a multiple eigenvalue or a set of nearly multiple eigenvalues. G. H. Golub & J. H. Wilkinson [1] and Bo Kågström & A. Ruhe [3] did not treat this problem directly but they tried to compute the invariant subspaces corresponding to clusterd eigenvalues. The fact that the eigenvalues are close enough is assured there by some measure essentially the singular value decomposition (say). Concerning this problem, S. M. Rump yields preferable results in [4] with exceptionally poor results for defective eigenvalues.

In this paper we propose a method to distinguish a multiple eigenvalue from nearly multiple eigenvalues. In section 2, we consider powers of a matrix as a submatrix for the invariant subspace corresponding to clustered eigenvalues. In section 3, we state representation formula of the eigen projection and eigen nilpotents. We propose in section 4 our method as an application of the results of preceding two sections and the results of numerical experiments for several kinds of matrices are shown. Some concluding remarks are in the last section.

2. Powers of the matrix with eigenvalues whose magnitudes are less than 1

Let $G = (g_{ij})$ be a non symmetric $p \times p$ matrix with eigenvalues $\{\lambda_j\}_{j=1}^p$ and $\|G\|_2 = 1$. Put $\bar{\lambda} = \frac{1}{p}\sum_{j=1}^p \lambda_j$ and let $\delta = \max_j |\lambda_j - \bar{\lambda}| < 1$. Then the following proposition holds.

Proposition $\forall \vec{z} \in \mathbf{C}^p$ with $\|\vec{z}\|_2 = 1$,

$$\|(G-\bar{\lambda}I)^p \vec{z}\|_2 = \begin{cases} 0 & \text{if } \lambda_j = \bar{\lambda},\ j = 1, 2, \cdots, p, \\ & (\bar{\lambda} \text{ is a multiple eigenvalue.}) \\ O(\delta^2) & \text{otherwise.} \end{cases}$$

Remark. We can say that if δ is small enough, that is, $\delta^2 \approx m(\varepsilon)$ (machine epsilon), then G looks like a Jordan block of height p under finite precision arithmetic, if $\|(G - \bar{\lambda}I)^{p-1}\vec{z}\|_2 \approx 1$.

Proof. Let $R = U^*GU$ with a unitary U by the Schur theorm:

$$R = \begin{pmatrix} \lambda_1 & r_1 & * & \dots & * \\ 0 & \lambda_2 & r_2 & \dots & * \\ \vdots & \vdots & \ddots & \dots & * \\ 0 & 0 & \dots & \lambda_{p-1} & r_{p-1} \\ 0 & 0 & \dots & 0 & \lambda_p \end{pmatrix} = \Lambda + N,$$

with diagonal matrix Λ and nilpotent N, that is, $N^p = 0$. We put $(R - \bar{\lambda}I)^p = ((\Lambda - \bar{\lambda}I) + N)^p = \sum_{k=0}^{p} T_k$, where each T_k is the sum of terms which are the products of k $(\Lambda - \bar{\lambda}I)$'s and $p - k$ N's.

Obviously $T_0 = N^p = 0$. Put $\Lambda - \bar{\lambda}I = \bar{\Lambda}$ then

$$T_1 = \bar{\Lambda}N^{p-1} + N\bar{\Lambda}N^{p-1} + \cdots + N^{p-2}\bar{\Lambda}N + N^{p-1}\bar{\Lambda}$$
$$= \begin{pmatrix} 0 & \dots & \dots & 0 & t_{1p} \\ 0 & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix},$$

where $t_{1p} = \sum_{j=1}^{p}(\lambda_j - \bar{\lambda})r_1 r_2 \cdots r_{p-1}$ is derived by a manipulation. Since $\sum_{j=1}^{p}(\lambda_j - \bar{\lambda}) = 0$, we have $T_1 = 0$. From the defintion of T_k, we have $\|T_k\| = O(\delta^k)\|N\|^{p-k}$ for $k \geq 2$. Thus $\|(R - \bar{\lambda}I)^p\| = \sum_{k=2}^{p} O(\delta^k)\|N\|^{p-k} = O(\delta^2)\|N\|^{p-2}$, provided that $\delta < \|N\|$. Hence $\|(G - \bar{\lambda}I)^p\vec{z}\|_2 = \|(U^*GU - \bar{\lambda}I)^p\vec{z}\|_2 = \|(R - \bar{\lambda}I)^p\vec{z}\|_2 = O(\delta^2)\|N\|^{p-2}$. Here the norm of a matrix is the operator norm on the complex Euclidean space $\mathbf{C}^p$. QED.

3. Representation formula of the eigen projection and eigen nilpotents

Let $A = (a_{ij})$ be a non symmetric $n \times n$ matrix with the set of eigenvalues $\sigma = \{\lambda_j,\, j = 1, \cdots, n\}$. Let $\sigma_1 = \{\lambda_j,\, j = 1, \cdots, p\}$ be an isolated cluster of eigenvalues in the complex plane. Put $\bar{\lambda} = \frac{1}{p}\sum_{j=1}^{p}\lambda_j$ and let $\delta = \max\limits_{0 \leq j \leq p} |\lambda_j - \bar{\lambda}| < 1$. Put $\sigma_2 = \sigma \setminus \sigma_1$. Let $R = \min\limits_{\lambda_j \in \sigma_1, \lambda_k \in \sigma_2} |\lambda_j - \lambda_k| \gg \delta$. Let C be a circle with center $\bar{\lambda}$ and radius r in the complex plane which encloses all elements of σ_1. We will take r such that $2\delta < r < \dfrac{R}{2}$ afterall. Here we often set $\lambda = \bar{\lambda}$ but it does not necessary for our theory as is seen in [5].

It is known that the following integrals give a projection on the invariant subspace corresponding to eigenvalues in σ_1. The integral is called the eigen projection if $k = 0$, and they are called eigen nilpotents for $k \geq 1$ in the case that $\bar{\lambda}$ is a multiple eigenvalue. (c.f T. Kato [2])

$$D_{\bar{\lambda}}^k \vec{z} = \frac{-1}{2\pi}\oint_C (\zeta - \bar{\lambda})^k (A - \zeta I)^{-1}\vec{z}\, d\zeta\ , \quad \text{for } \vec{z} \in \mathbf{C}^n$$

and $k = 0, 1, 2, \cdots, p-1$.

$D_{\bar{\lambda}}^k$ satisfy the chain rule: $(A - \bar{\lambda}I)D_{\bar{\lambda}}^k \vec{z} = D_{\bar{\lambda}}^{k+1}\vec{z}$ for $k = 1, \cdots, p-1$. Here $D_{\bar{\lambda}}^p \vec{z} = 0$, that is, $D_{\bar{\lambda}}^{p-1}\vec{z}$ is the eigenvector corresponding to $\bar{\lambda}$. We denote by P_C the eigen projection $D_{\bar{\lambda}}^0$.

Even if σ_1 is not a multiple eigenvalue, P_C is also the projection on the invariant subspace corresponding to eigenvalues in σ_1, that is, $P_C\vec{z}$, which is independent of $\bar{\lambda}$ but depends on the circle C, is a vector in the invariant subspace corresponding to eigenvalues in σ_1.

Corresponding to them, we define the following numerical integrations: Let $\{\mu_j,\, j = 0, \cdots m-1\}$ be m points cyclotomically shifted on C. We use here λ' instead of $\bar{\lambda}$ as a more general parameter.

$$\hat{D}_{\lambda'}^k \vec{z} = \frac{-r}{m}\sum_{j=0}^{m-1} e^{i\theta j}(\mu_j - \lambda')^k (A - \mu_j I)^{-1}\vec{z},$$

$$\hat{P}_C\vec{z} = \hat{D}_{\lambda}^0 \vec{z} = \frac{-r}{m}\sum_{j=0}^{m-1} e^{i\theta j}(A - \mu_j I)^{-1}\vec{z}.$$

where $\theta_j = \frac{2\pi j}{m},\ j = 0, \cdots m-1$.

$\hat{D}_{\lambda'}^k \vec{z}$ is a linear combination of $(A - \mu_j I)^{-1}\vec{z},\ j = 1, 2, \cdots, m-1$. λ' is a parameter independent of λ. $\hat{P}_C\vec{z}$ is given from the values $(A - \mu_j I)^{-1}\vec{z}$ at m points on C.

The following two results are known [5]. Here, we use the notation λ_h instead of $\bar{\lambda}$ if σ_1 is a multiple eigenvalue.

1. $\|\hat{D}_{\lambda_h}^k \vec{z} - D_{\lambda_h}^k \vec{z}\| = O\left(\left|\dfrac{r}{R}\right|^{m-k}\right) \quad (0 \leq k < m)$, where $D_{\lambda_h}^k \vec{z} = 0\ (p \leq k)$, provided $R > r$.

2. Putting $\vec{u} = \hat{D}_{\lambda'}^{p-1}\vec{z}$ and $\tilde{\lambda} = (A\vec{u}, \vec{u})/(\vec{u}, \vec{u})$, we have

$$\left|\lambda_h - \frac{\tilde{\lambda} + (p-1)\lambda'}{p}\right| = O(|\lambda_h - \lambda'|^2).$$

Taking $\lambda'' = \frac{\tilde{\lambda} + (p-1)\lambda'}{p}$ as the new λ', we can derive an iterative process using $\hat{D}_{\lambda'}^{p-1}\vec{z}$, which yields a convergent sequence $\{\lambda''\}$ to λ_h of order 2.

Note that the accuracy of these values depends on those of $(A - \mu_j I)^{-1}\vec{z}, j = 0, \ldots, m-1$.

We have the following theorem of representation formulas for those numerical integrations. Though it was proved including some other properties in the unpublished private note of S.T. Kuroda, we give here an elementary proof of it.

Theorem (The formulation of this theorem is due to S.T. Kuroda.)

$$\hat{P}_C\vec{z} = \left[I - \left(\frac{A - \lambda I}{r}\right)^m\right]^{-1}\vec{z}, \tag{1}$$

$$\hat{D}_{\lambda'}^k \vec{z} = (A - \lambda' I)^k \hat{P}_C\vec{z}. \tag{2}$$

Proof. Since

$$\hat{D}_{\lambda'}^k = \frac{-r}{m}\sum_{j=0}^{m-1} e^{i\theta j}(\mu_j - \lambda')^k (A - \mu_j I)^{-1}, \quad \text{and}$$

$$\hat{P}_C = \hat{D}_{\lambda}^0 = \frac{-r}{m}\sum_{j=0}^{m-1} e^{i\theta j}(A - \mu_j I)^{-1},$$

with $\theta_j = \frac{2\pi j}{m}$, putting $w = e^{i\theta}$ and $\nu = \lambda - \lambda'$, we have

$$\begin{aligned}\hat{D}_{\lambda'}^k &= \frac{-r}{m}\sum_{j=0}^{m-1} w^j(\mu_j - \lambda')^k (A - \mu_j I)^{-1} \\ &= \frac{r}{m}\sum_{j=0}^{m-1}[(\lambda - \lambda') + rw^j]^k w^j (rw^j I - (A - \lambda I))^{-1}\end{aligned}$$

$$= \frac{1}{m}\sum_{j=0}^{m-1}[\nu+\tau w^j]^k (I-\frac{A-\lambda I}{r}w^{-j})^{-1}$$
$$= \frac{1}{m}\sum_{j=0}^{m-1}\left\{\sum_{s=0}^{k}\binom{k}{s}\nu^{k-s}(rw^j)^s\right\}$$
$$\times\left\{\sum_{t=0}^{m-1}\left[\left(\frac{A-\lambda I}{r}\right)w^{-j}\right]^t\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}\right\}$$
$$= \frac{1}{m}\sum_{t=0}^{m-1}\sum_{s=0}^{k}\left[\binom{k}{s}\left(\frac{A-\lambda I}{r}\right)^t r^s\nu^{k-s}\sum_{j=0}^{m-1}w^{js}w^{-jt}\right]$$
$$\times\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}$$
$$= \sum_{s=0}^{k}\binom{k}{s}\left(\frac{A-\lambda I}{r}\right)^s r^s\nu^{k-s}\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}$$
$$= [(A-\lambda I)+\nu I]^k\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}$$
$$= [(A-\lambda I)+\lambda I-\lambda' I]^k\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}$$
$$= (A-\lambda' I)^k\left[I-\left(\frac{A-\lambda I}{r}\right)^m\right]^{-1}.$$

To put $k=0$ yields (1) and (2) is obvious.

4. Applications and numerical experiments

Because $G = AP_C$ is the restriction of A on the invariant subspace corresponding to the eigenvalues in σ_1, the proposition in section 2 holds for AP_C. If we take m points properly, $A\hat{P}_C$ is a good approximation of $G = AP_C$. Then the $\hat{D}_{\bar\lambda}^k$'s are also good approximations of $(G-\bar\lambda I)^k$, $k = 1, 2, \cdots$. Note that if $q < p$ then $(G-\bar\lambda I)^q = N^q + O(\delta)N^{q-1} + O(\delta^2)N^{q-2} + \cdots$.

4.1. Two assertions on the problem

According to the previous theorem, we can apply the preceding proposition to the matrix which operates on the invariant subspace corresponding to σ_1. Then we have the following result on the problem of how we can conclude σ_1 to be a multiple eigenvalue or nearly multiple eigenvalues under finite precision arithmetic.

Let ε be a certain number which should be considered to be as small enough as zero under the computational circumstances. We have the following two conclusions assuming that a number τ is 0 if $\tau \le \varepsilon$.

The first one is mathematically exact.

Assertion 1. If $\|\hat{D}_{\bar\lambda}^p\vec{z}\| = \gamma > 0$ with small γ (that is $\|\hat{D}_{\bar\lambda}^p\vec{z}\| = \gamma > \varepsilon$) provided that $\|\hat{D}_{\bar\lambda}^{p-1}\vec{z}\| = O(1)$, then σ_1 is a cluster of nearly multiple eigenvalues with its radius $\ge \sqrt{\gamma}$.

We propose the following second statement to be used in practical cases of general matrices, considering the applicability after observing parameters and the resulting data.

Assertion 2. If $\|\hat{D}_{\bar\lambda}^p\vec{z}\| = 0$, that is $\|\hat{D}_{\bar\lambda}^p\vec{z}\| \le \varepsilon$, and $\|\hat{D}_{\bar\lambda}^{p-1}\vec{z}\| = O(1)$ then σ_1 is a multiple eigenvalue with error less than $\sqrt{\varepsilon}$. Moreover if $p \ge 2$ then it is defective of height p.

For example, if every computation is carried with errors of the order $O(10^{-d})$, then we conclude that "the resulting value is zero" is true within error less than $O(10^{-\frac{d}{2}})$.

Remark. Assertion 2 does not necessarily stand for all cases. There are some exceptional special cases. For example, if $\lambda_j = \lambda + t\exp(\frac{2\pi i}{p}j)$, $j = 0, 1, \cdots, p-1$ then $\|\hat{D}_{\bar\lambda}^p\vec{z}\| = O(t^p)$ holds. It is known that a matrix A with a defective eigenvalue λ of multiplicity p is perturbed with a small ε and a certain F into $A+\varepsilon F$, then the defective eigenvalue λ splits into $\lambda_j = \lambda + (\varepsilon)^{\frac{1}{p}}\exp(\frac{2\pi i}{p}j)$, $j = 0, 1, \cdots, p-1$. In this case the error bound should be expressed to be $O(10^{-\frac{d}{p}})$. See Example 5 in **4.2**

4.2. Numerical experiments for 4.1

In [4], S.M. Rump proposed a method which can be applied to compute the Jordan canonical form. It is implemented in Matlab and yields accurate eigenvalues. But it is not satisfactory for defective eigenvalues of height greater than 2, although the mean of clusterd eigenvalues is remarkably accurate.

It can be concluded theoretically that the accuracy of the numerical computation of our method depends on that of the linear equations $(A-\mu_j I)\vec{w} = \vec{z}$. We show numerical results with the condition numbers of linear equations in each experiment.

In the test computation, we used Matlab to compute the mean $\bar\lambda$ of the eigenvalues in σ_1 and to solve m linear equations. Though Matlab yields poor values for defective eigenvalues, $\bar\lambda$ has a good accuracy.

Test matrices and vectors:

Let J be a matrix with a certain required form. Using a random nonsingular matrix X, A is defined by $A = XJX^{-1}$ in our experiments. Most of our numerical experiments were carried out for the 100×100 matrix A.

The components of the initial vector z are taken randomly with its magnitude ≤ 1.

Our first example is to certify the assertion 1.

Example 1. Matrices with nearly multiple k eigenvalues.

Matrix: $n = 100$, $k = 4$, $\sigma_1 = \{\lambda_j = 2.0a + t\exp(i\frac{2\pi}{9}j)(j = 0, 1, 4, 5)\}$ and all the other eigenvalues

are within the unit circle in the complex plane. The submatrix of J corresponding to the eigenvalues in σ_1 is the 4×4 matrix such that $\lambda_1, \cdots, \lambda_4$ are diagonal, the super diagonal components are 1 and all the others are 0.

The condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$: max= 7.03×10^4, min= 5.94×10^4, mean= 6.31×10^4. The values of $\|\hat{D}_{\hat{\lambda}}^0 z\|$, $\|\hat{D}_{\hat{\lambda}}^1 z\|$ and $\|\hat{D}_{\hat{\lambda}}^2 z\|$ for each t are the

Table 1. norms of principal vectors $(k = 4)$

t	$\|\hat{D}_{\hat{\lambda}}^3 z\|$	$\|\hat{D}_{\hat{\lambda}}^4 z\|$	$\|\hat{D}_{\hat{\lambda}}^5 z\|$
10^{-3}	3.560	5.907×10^{-6}	5.315×10^{-6}
10^{-4}	3.562	5.910×10^{-8}	5.317×10^{-8}
10^{-5}	3.562	5.911×10^{-10}	5.317×10^{-10}
10^{-6}	3.562	5.910×10^{-12}	5.330×10^{-12}
10^{-7}	3.562	9.654×10^{-14}	5.621×10^{-14}
0	3.562	8.979×10^{-14}	2.226×10^{-14}

same, that is, approximately equal to 6.46, 6.25 and 3.96 respectively. So we omitted them from Table 1.

Note in Table 1 that $\|\hat{D}_{\hat{\lambda}}^4 z\| = O(t^2)$ and $\|\hat{D}_{\hat{\lambda}}^l z\| = O(1)$, $l = 0, 1, 2, 3$. As for the cases $t = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$ we conclude that σ_1 is not a multiple eigenvalue. From the values of $\|\hat{D}_{\hat{\lambda}}^3 z\|$ and $\|\hat{D}_{\hat{\lambda}}^4 z\|$, we can decide that the diameter of the cluster is of order $O(t)$.

The second example is the case of a multiple eigenvalue. It shows that our method works well for numerical computation of the Jordan canonical form.

Example 2. Matrix with a k-fold defevtive eigenvalue.(cf. S. M. Rump[4]-Table 9, test matrix.)

Matrix: Let 2 be a multiple and defective eigenvalue of A. Let J be a canonical form of the 100×100 matrix with the Jordan block corresponding to 2 and all the other eigenvalues are within the unit circle on the complex plane.

Parameters: $m = 24$ and $r = 2^{-3} = 0.125$.

Using the function `eig` in MATLAB, we get k approximated eigenvalues. $\bar{\lambda}$ is computed from them.

Table 2. Condition number of $A - \mu_j I$

k	Max.	Min.	Avg.
2	1.68×10^4	1.68×10^4	1.68×10^4
3	1.30×10^5	1.12×10^5	1.21×10^5
4	1.61×10^6	1.22×10^6	1.40×10^6
5	3.05×10^7	2.54×10^7	2.71×10^7

Because the values of $\|\hat{D}_{\hat{\lambda}}^0 z\|$'s for each k are O(1), they are omitted from Table 3.

We conclude in Table 3 that $\|\hat{D}_{\hat{\lambda}}^k z\| = 0$ by our theory. Combining with the fact that $\|\hat{D}_{\hat{\lambda}}^l z\| = O(1)$, $l = 0, \cdots, k-1$, we can conclude that the eigenvalue 2 is a defective eigenvalue of height k.

Table 3. norms of principal vectors ($m = 24$**)**

k	$\|\hat{D}_{\hat{\lambda}}^1 z\|$	$\|\hat{D}_{\hat{\lambda}}^2 z\|$	$\|\hat{D}_{\hat{\lambda}}^3 z\|$	$\|\hat{D}_{\hat{\lambda}}^4 z\|$	$\|\hat{D}_{\hat{\lambda}}^5 z\|$
2	4.433	2.6E-14	8.6E-15	1.4E-16	1.2E-17
3	2.140	1.0E-1	1.3E-14	3.5E-15	1.9E-16
4	2.249	1.911	7.8E-2	3.0E-14	1.1E-14
5	10.96	6.060	2.790	2.158	3.6E-13

Next example is the cases of various size of matrices. It shows that our method does not depend on the size but on the condition number of matrices.

Example 3. Matrices of various dimensions. (Rump[4]-Table 7, test matrix.)

This example comprises a 10-fold eigenvalue 2 in five Jordan blocks each of size 2, for different dimensions n. Furthermore, the matrix is generated to have one eigenvalue 1 and $n-11$ randomly distributed eigenvalues within $[-1, 1]$

Parameters: $m = 24$ and $r = 2^{-3} = 0.125$.

Using the function `eig` in MATLAB, we get 10 approximated eigenvalues. $\bar{\lambda}$ is computed from them.

The condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$ were independent of t. (See Table 4).

Table 4. Condition number of $A - \mu_j I$

n	Max.	Min.	Avg.
50	6.15×10^5	5.37×10^5	5.74×10^5
100	3.20×10^8	2.86×10^8	3.05×10^8
200	9.88×10^4	9.24×10^4	9.57×10^4
500	1.25×10^6	1.19×10^6	1.22×10^6

Table 5. norms of principal vectors ($m = 24$**)**

n	$\|\hat{D}_{\hat{\lambda}}^0 z\|$	$\|\hat{D}_{\hat{\lambda}}^1 z\|$	$\|\hat{D}_{\hat{\lambda}}^2 z\|$	$\|\hat{D}_{\hat{\lambda}}^3 z\|$
50	54.83	39.51	4.6E-12	2.2E-12
100	1121	1021	4.0E-8	1.7E-8
200	18.46	12.49	1.9E-13	9.3E-14
500	17.59	8.458	4.6E-13	2.5E-13

The norms of the principal vectors computed by our method for various size n are seen in Table 5. They show that the efficiency of our method depends only on the condition numbers of $A - \mu_j I$, $j = 0, \cdots m-1$.

In example 4 we can see the norms of eigen nilpotent vectors for non defective matrices.

Example 4. Non-defective matrices with a cluster of k-eigenvalues.

Matrix: $n = 100$, $k = 5$. $\sigma_1 = \{\lambda_j = 2.0 + t\exp(i\frac{2\pi}{5}j), j = 0, \ldots, 4\}$ and the remaining 95 eigenval-

ues are randomly distributed in the unit disk on the complex plane.

Parameters $m = 24$ and $r = 2^{-3} = 0.125$.
The condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$:
max= 2.97×10^3, min= 2.63×10^3, mean= 2.79×10^3.

Table 6. norms of principal vectors

t	$\|\hat{D}_\lambda^0 z\|$	$\|\hat{D}_\lambda^1 z\|$	$\|\hat{D}_\lambda^2 z\|$	$\|\hat{D}_\lambda^3 z\|$	$\|\hat{D}_\lambda^4 z\|$
10^{-3}	4.7	4.4E-3	4.5E-6	4.5E-9	4.4E-12
10^{-4}	4.7	4.4E-4	4.5E-8	4.5E-12	(10^{-16})
10^{-5}	4.7	4.4E-5	4.5E-10	4.5E-15	(10^{-18})
10^{-6}	4.7	4.4E-6	4.5E-12	(10^{-17})	(10^{-18})
10^{-7}	4.7	4.4E-7	4.5E-14	(10^{-17})	(10^{-18})
10^{-8}	4.7	4.4E-8	(10^{-16})	(10^{-17})	(10^{-18})
0	4.7	2.1E-14	(10^{-16})	(10^{-17})	(10^{-18})

These values for simple eigenvalues could be explained by the representation formula for eigen-projection.

Note that $\|\hat{D}_\lambda^0 z\| = O(1)$ and $\|\hat{D}_\lambda^l z\| = O(t^l)$, $l = 1, 2, \cdots$.(See Table 6.)

It is sure, of course that there are exceptional cases for which Assertion 2 does not stand. There exists a well known example that a small(ε-) perturbation to the matrix with a defective eigenvalue of multiplicity p causes the multiple eigenvalue to split into p eigenvalues on a circle of radius $\varepsilon^{\frac{1}{p}}$. (cf. Remark followed to Assertion 2.) The following Example 5 is one of them, that is, an example answering if a cluster of cyclotomically shifted p eigenvalues on a circle can be decided to be separated ones. The conclusion is that if low precision arithmetic is employed in numerical computations then even eigenvalues distributed on a circle with rather large radius are hardly distinguished: but the fact that they are not a multiple eigenvalue is revealed gradually as we employ higher precision arithmetic.

Example 5. Perturbed matrix with a defective eigenvalue of multiplicity k.

Matrix: $n = 100$! %$\sigma_1 = \{\lambda_j = 2.0 + t\exp(i\frac{2\pi}{5}j), j = 0, \ldots, k-1\}$ and all the other eigenvalues are apart from 2.0. Parameters: $m = 24$! $r = 2^{-2} = 0.25$. The results of the case of $k = 5$ is seen in Table 7. Condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$:
max =2.65×10^5, min = 2.44×10^3, mean = 2.54×10^5.

The values of $\|\hat{D}_\lambda^0 z\|$ and $\|\hat{D}_\lambda^1 z\|$ are almost the same for each t, that is, 4.70 and 3.69, respectively. So they are omitted from Table 7.

We can find no differences in Table 7 among values for various t's because the double precision arithmetic has too low precision to cover such a value $(10^{-3})^5 = 10^{-15}$. In this case, according to Assertion 2, we conclude that $\lambda = 2$ is a multiple and defective eigenvalue of height 5.

The results of the case of $k = 4$ is seen in Table 8.

Table 7. norms of principal vectors $(k = 5)$

t	$\|\hat{D}_\lambda^2 z\|$	$\|\hat{D}_\lambda^3 z\|$	$\|\hat{D}_\lambda^4 z\|$	$\|\hat{D}_\lambda^5 z\|$
10^{-3}	3.510	3.366	2.748	2.5E-13
10^{-4}	3.512	3.368	2.749	2.9E-13
10^{-5}	3.513	3.368	2.749	2.1E-13
10^{-6}	3.513	3.368	2.749	3.7E-13
10^{-7}	3.513	3.368	2.749	1.9E-13
0	3.513	3.368	2.749	2.2E-13

Condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$:
max = 7.03×10^4, min = 5.94×10^4, mean = 6.31×10^4.

Table 8. norms of principal vectors $(k = 4)$

t	$\|\hat{D}_\lambda^3 z\|$	$\|\hat{D}_\lambda^4 z\|$	$\|\hat{D}_\lambda^5 z\|$
10^{-3}	3.561	6.4E-12	6.2E-12
10^{-4}	3.562	4.2E-14	1.8E-14
10^{-5}	3.562	1.9E-14	1.5E-14
10^{-6}	3.562	1.2E-13	1.9E-14
10^{-7}	3.562	7.0E-14	2.1E-14
0	3.562	8.9E-14	2.2E-14

The values of $\|\hat{D}_\lambda^0 z\|$, $\|\hat{D}_\lambda^1 z\|$ and $\|\hat{D}_\lambda^2 z\|$ are almost the same for each t, that is, 6.46, 6.25 and 3.95, respectively. So they are omitted from Table 8.

Note that, in the case of $t = 10^{-3}$, $\|\hat{D}_\lambda^l z\| = O(t^4)$, $l = 4, 5$ and $\|\hat{D}_\lambda^3 z\| = O(1)$. According to Assertion 1, we conclude here that the diameter of the set of the nearly multiple eigenvalues is greater than $\sqrt{\|\hat{D}_\lambda^4 z\|}(\approx 10^{-6})$. It is not so accurate but the conclusion is true.

The results of the case of $k = 3$ is seen in Table 9.

Condition numbers of $A - \mu_j I, j = 0, \ldots, m-1$:
max = 9.03×10^3, min = 8.33×10^3, mean = 8.59×10^3.

Table 9. norms of principal vectors $(k = 3)$

t	$\|\hat{D}_\lambda^2 z\|$	$\|\hat{D}_\lambda^3 z\|$	$\|\hat{D}_\lambda^4 z\|$	$\|\hat{D}_\lambda^5 z\|$
10^{-3}	1.587	1.9E-9	1.8E-9	1.5E-9
10^{-4}	1.587	1.9E-12	1.8E-12	1.5E-12
10^{-5}	1.587	9.1E-15	1.0E-15	9.3E-16
10^{-6}	1.587	6.1E-15	2.5E-15	1.7E-15
10^{-7}	1.587	1.0E-14	2.4E-15	1.24E-15
0	1.587	3.6E-15	2.0E-15	1.5E-15

The values of $\|\hat{D}_\lambda^0 z\|$ and $\|\hat{D}_\lambda^1 z\|$ are almost the same for each t, that is, 1.9 and 1.8, respectively. So they are omitted from Table 9.

Note that, in the cases of $t = 10^{-3}$ and $t = 10^{-4}$, $\|\hat{D}_\lambda^l z\| = O(t^3)$, $l = 3, 4, 5$ and $\|\hat{D}_\lambda^l z\| = O(1)$, $l =$

$0, 1, 2$. According to Assertion 1, we conclude here that the diameter of the set of the nearly multiple eigenvalues is greater than $\sqrt{\|\hat{D}_{\hat{\lambda}}^3 z\|}(\approx 10^{-4.5}, 10^{-6})$. It is not so accurate but the conclusion is true.

From Example 5, it seems that the small perturbation of order of machine epsilon for defective eigenvalues does not affect to this eigen value problem in our method, because such a perturbed defective eigenvalue is computed as a defective eigenvalue itself by Assertion 2. Nearly multiple eigenvalues would be distinguished using proper high precision arithmetic computation by Assertion 1.

4.3. Concluding remarks

1. A method how to decide a cluster of eigenvalues to be multiple or nearly multiple is proposed. The numerical experiments for some kinds of matrices show that the method corresponding to Assertion 1 works well theoretically.
2. Though the test matrices are made by $A = XJX^{-1}$ with a random nonsingular matrix X, the test matrices may be well posed in the sense that the off diagonal components of R derived by Schur theorem are of order $O(1)$. So our test computation seems to show the high efficiency of our method. We know, of course, that there are many matrices that cannnot be treated by our method.
3. Considering the results of Example 5 and the following remark, one may expect by pursuing along this way to get the method to know how long the precision of arithmetic is required to conclude that a cluster of eigenvalues is a multiple eigenvalue within the prerequisite error bound.
4. In order to see the limit of our method we should try more experiments for various ill-conditioning examples to observe what kinds of data characterize the ill-conditioning of the matrices in the practical numerical computation. And we need more theoretical investigation.

References

[1] Golub, G. H. and Wilkinson, J. H., Ill-conditioned eigensystems and the computation of the Jordan canonical form, SIAM Review,18 (1976), 578-619.

[2] Kato, T., *Perturbation Theory*, Springer Verlag, New York, second edition, 1976.

[3] Kågström, Bo and Ruhe, A., An Algorithm for Numerical Computation of the Jordan Normal Form of a Complex Matrix, ACM Trans. Math. Software, Vol. 6, No.3 (1980),398-429

[4] Rump, S. M. Computational error bounds for multiple or nearly multiple eigenvalues, Linear Algebra and its Applications 324 (2001), 209-226.

[5] Suzuki, T. and Watanabe, E., Numerical computation of the Jordan canonical form, Proc. The Third IMACS International Symposium on Iterative Method in Scientific Computation, Jackson Hole, Wyoming, USA, IMACS Series in Computational and Applied MAthematics. Vol.4 (1998),65-70.

[6] Suzuki, T. and Suzuki, T., An eigenvalue problem for derogatory matrices, J. Computational and Applied Mathematics 199 (2007), 245-250.

The Fundamental Theorems of Interval Analysis

M.H. van Emden and B. Moa
Department of Computer Science
University of Victoria, Canada

Abstract

Expressions are not functions. Confusing the two concepts or failing to define the function that is computed by an expression weakens the rigour of interval arithmetic. We give such a definition and continue with the required re-statements and proofs of the fundamental theorems of interval arithmetic and interval analysis.

1 Introduction

> *Make things as simple as possible, but not simpler.*
>
> Albert Einstein.

The *raison d'être* of interval arithmetic is *rigour*. Yet it appears that the most fundamental fact, sometimes referred to as the "Fundamental Theorem of Interval Arithmetic", is not rigorously established. The fact in question can be described as follows.

Let e be an expression with $\langle x_1, \ldots, x_n \rangle$ as an ordered set of variables (i.e. a finite sequence of distinct variables). Let f be the function in $\mathcal{R}^n \to \mathcal{R}$ that is computed by e. Let the result of evaluating e with intervals $I_1, \ldots, I_n$ substituted for $x_1, \ldots, x_n$ be an interval Y. Then

$$\{f(x_1, \ldots, x_n) \mid x_1 \in I_1, \ldots, x_n \in I_n\} \subset Y \quad (1)$$

Although this fact is fundamental to everything that is done in interval arithmetic, we have failed to find in the literature a definition of what it means for *an expression to compute a function*. In Section 1.2 we review the literature that we consulted.

In (1) the interval Y is typically considerably wider than the range of function values. Interval analysis relies on the fact that, as $I_1, \ldots, I_n$ become narrower, the sides in (1) become closer to each other. A theorem to this effect, such as 2.1.1 or 2.1.3 in [9] deserves to be called Fundamental Theorem of Interval Analysis rather than interval arithmetic.

Both theorems should rest on the foundation provided by a definition of the function computed by an expression. We give such a definition for sets; as intervals are sets, the definition applies to intervals as a special case.

1.1 Expressions and functions

An expression is an entity consisting of *symbols*; it is an element of a formal language in the sense of computer science. Some of these symbols denote operations; others are constants or variables and denote reals or intervals, according to the chosen interpretation.

Unlike an expression, a numerical function is an element of the function space $\mathcal{R}^n \to \mathcal{R}$, for a suitable positive integer n. Variables only occur in expressions, where they can re-occur any number of times. Variables do not occur in functions; in fact, the notion of "occurs in" is not applicable to functions in $\mathcal{R}^n \to \mathcal{R}$. Instead, a function in $\mathcal{R}^n \to \mathcal{R}$ is a map from n-tuples of reals to reals; the elements of the n-tuples are properly called *arguments*, rather than "variables".

An additional reminder of the need to distinguish between expressions and functions is that different expressions can compute the same function. Yet another reminder is that there exist functions that are not computable, whereas all expressions are, like programs, instructions for computations. Contrary to programs in general, expressions of the type of interest to interval arithmetic can be evaluated in finite time. Hence the functions computed by these expressions belong to the computable subset of functions.

Of course, "the set of expressions" could be made precise by means of a formal grammar. For the purpose of this paper, it is sufficient to define an expression as follows.

1. A variable is an expression.
2. If E is an expression and if φ is a unary operation symbol, then φE is an expression.
3. If E_1 and E_2 are expressions and if $\diamond$ is a binary operation symbol, then $E_1 \diamond E_2$ is an expression.

To make the definition formal, we would have to spell out the appearance of the variables and of the operation symbols.

In whatever way expressions are defined, the resulting set is disjoint from the set $\mathcal{R}^n \to \mathcal{R}$, whatever n is. What is needed to turn the Fundamental Fact (1) into a theorem is to define "function computed by an expression" as a mapping from the set of expressions in n variables to $\mathcal{R}^n \to \mathcal{R}$. As observed above, this mapping is neither injective nor surjective. This mapping can be called the *semantics* of the language of expressions.

1.2 Remarks on the literature

Moore [8] avoids the problem of defining the function defined by an expression by not making the distinction. As explained in the previous section, this is not correct. Jaulin *et. al.* [6], Theorem 2.2, assume that the problem is taken care of by composition of functions, but make unjustified simplifications. Composition is indeed a promising approach, which we will pursue in Section 4.

Neumaier [9] does distinguish between expressions and functions, but the expressions as he defined them fail to be computable. In fact, following the definition he gave in page 13, every real number is an element of the set of arithmetical expressions. The simplicity arises from the fact that all real numbers are defined as (sub)expressions. This introduces infinite expressions: whatever notation is chosen for the reals, most (in the sense of a subset of measure one) are infinite. In this way effective computability is lost.

Moreover, Neumaier starts with an arithmetic expression f, and then defines the interval evaluation of f, which he denotes by the same symbol f. To deal with partial functions, he introduced a *NaN* symbol, and the results of operations on this symbol. He then defined the restriction of f to its real domain $D_f = \{x \in \mathcal{R}^n \mid f(x) \neq NaN\}$ to be the real evaluation of f. We do not see the need for this indirect approach: partial functions are a perfectly natural and hardly novel generalization of functions that are total.

Ratschek and Rokne also distinguish expressions from functions. In [12] they refer to their earlier book [11] for a definition. This is a mistake, because on page 23, after a heuristic discussion of the connection between expression and functions, they refer to texts in logic and universal algebra for a definition. However, these assume that all functions are total. This is not always the case for the expressions of interest to interval arithmetic; consider for example $\sqrt{x}$. As only a few exotic varieties of logic allow function symbols to be interpreted by partial functions, it is better for interval arithmetic to use set theory as basis for its fundamental theorems. In fact, these exotic varieties are subject to considerable controversy [2, 10], so not suitable as a fundament for interval analysis.

2 Set theory preliminaries

This section establishes the concepts, terminology and notation for this paper. It is necessary because the present investigation is unusual in that all functions are what are usually called "partial functions". To avoid having to qualify with "partial" every time a function is mentioned, we define "function" to mean what is usually referred to as "partial function". In other respects, we adhere closely to standard expositions of set theory, such as [3, 1] and standard introductions such as found in authoritative texts such as [7].

Definition 1 *A* function f *consists of a* source*, a* target*, and a* map*. The source and target are sets. The map associates each element of a subset of the source with a unique element of the target.*

The set of functions with source S and target T is denoted by the term $S \to T$. If a function $f \in S \to T$ associates $x \in S$ with $y \in T$, then one may write y as $f(x)$. When only the association under f between x and y is relevant, we write $x \mapsto y$.

Example 1 *The square root is a function in $\mathcal{R} \to \mathcal{R}$ that does not associate any real with any negative real and associates with $x \in \mathcal{R}$ the unique non-negative $y \in \mathcal{R}$ such that $y^2 = x$ if $x \geq 0$.*

The term $f(x)$ is undefined if there is no $y \in T$ associated with $x \in S$ by $f \in S \to T$. We take $\{f(x) \mid x \in S\}$ to mean

$$\{y \in T \mid \exists x \in S \text{ and } f \text{ associates } y \text{ with } x\}.$$

That is, $\{f(x) \mid x \in S\}$ is defined even though $f(x)$ may not be defined for every $x \in S$.

Example 2 *$\{\sqrt{x} \mid x \in \mathcal{R}\}$ is defined and is the set of non-negative reals.*

$\{x/y \mid x \in \{1\}$ and $y \in \mathcal{R}\}$ is defined and is $\mathcal{R} \setminus \{0\}$.

The subset of S consisting of x with which $f \in S \to T$ associates a $y \in T$ is called the *domain* of f, denoted $\mathrm{dom}(f)$. If $\mathrm{dom}(f) = S$, then f is said to be a *total function*. $\{f(x) \mid x \in S\}$ is called the *range* of f. We introduced the unusual terms "source" for S and "target" for T because of the need to distinguish them from "domain" and "range".

Definition 2 *The set of functions with source S and target T is denoted $S \to T$ and is called a "type" or "function space".*

Again, this differs from the usual meaning of $S \to T$, where it only contains total functions. To say that f "is of type" $S \to T$ means that $f \in S \to T$.

Definition 3 *Let $f \in S \to T$ and $g \in T \to U$. The* composition $g \circ f$ *of f and g is the function in $S \to U$ such that $g \circ f$ associates $x \in S$ with $z \in U$ iff there exists a $y \in T$ such that f maps x to y and g maps y to z.*

This is the conventional definition of composition. It requires the target of f to be the same set as the source of g. Because of this requirement it is not clear what composition Jaulin *et. al.* have in mind in [6], Theorem 2.2.

It follows from Definition 3 that the domain of definition of $f \circ g$ is a subset of that of f.

Example 3 *$f \circ g \circ h$ has $\{0\}$ as domain if $f \in \mathcal{R} \to \mathcal{R}$ is such that it maps $x \mapsto \sqrt{x}$, $g \in \mathcal{R} \to \mathcal{R}$ is such that it maps $x \mapsto -x$, and $h \in \mathcal{R} \to \mathcal{R}$ is such that it maps $x \mapsto |x|$. In other words, $\sqrt{(-|x|)}$ is undefined for all $x \in \mathcal{R}$ except when $x = 0$.*

Let $f \in S \to T$. The elements of S are called "arguments" of f. Note that if a function associates an x in S with a y in T, it only so associates a *single* element of S. In that respect, all functions are "single-argument" functions. But S and T may be any sets whatsoever. Suppose $f \in \mathcal{R}^n \to \mathcal{R}$. Now the single elements in the source of f, the arguments of f, are n-tuples of reals. Thus we interpret the usual $f(x_1, \ldots, x_n)$ as $f(\langle x_1, \ldots, x_n \rangle)$.

Definition 4 *Let $f_1 \in S_1 \to T_1$ and $f_2 \in S_2 \to T_2$. The* Cartesian product *of f_1 and f_2, denoted $f_1 \times f_2$, is a function in $S_1 \times S_2 \to T_1 \times T_2$ having domain $dom(f_1) \times dom(f_2) = \{\langle x_1, x_2 \rangle \mid x_1 \in dom(f_1)$ and $x_2 \in dom(f_2)\}$, and mapping every $\langle x_1, x_2 \rangle$ in $dom(f_1) \times dom(f_2)$ to $\langle f_1(x_1), f_2(x_2) \rangle$.*

Definition 5 *Let f be a function in $S \to T$. Let F be a total function in $\mathcal{P}(S) \to \mathcal{P}(T)$. F is a* set extension *of f iff $\{f(x) \mid x \in X\} \subset F(X)$ for all subsets X of S. The total function in $\mathcal{P}(S) \to \mathcal{P}(T)$ with map $X \mapsto \{f(x) \mid x \in X\}$ is a set extension and is called the* canonical set extension *of f. We will use $f(D)$ to denote $\{f(x) \mid x \in D\}$.*

3. Intervals are sets — interval extensions are set extensions

As we saw, partial functions have set extensions that are total. This is of particular interest in numerical computation, where some important functions, such as division and square root, are not everywhere defined.

In some treatments of interval arithmetic this leads to the situation in which division by an interval containing zero is not defined. This is not necessary: if one regards an interval as a set and an interval extension as a set extension, then the interval extension is everywhere defined. This is the approach taken in [5], which will be summarized here.

A well-known fact is that the closed, connected sets of reals have one of the following forms: $\{x \in \mathcal{R} \mid x \leq b\}$, $\{x \in \mathcal{R} \mid a \leq x\}$, $\{x \in \mathcal{R} \mid a \leq x \leq b\}$, as well as $\mathcal{R}$ itself. Here a and b are reals. Note that the empty subset of $\mathcal{R}$ is an interval also, as no ordering is assumed between a and b.

The closed, connected sets of reals are defined to be the *real intervals*. They are denoted $[-\infty, b]$, $[a, \infty]$, $[a, b]$, and $[-\infty, \infty]$. These notations are just a shorthand for the above set expressions. They are not meant to suggest that, for example, $-\infty \in [-\infty, b] = \{x \in \mathcal{R} \mid x \leq b\}$. This is not the case because $[-\infty, b]$ is a set of reals and $-\infty$ is not a real.

The floating-point numbers are a set consisting of a finite set of reals as well as $-\infty$ and ∞. The real floating-point numbers are ordered as among the reals. The least (greatest) element in the ordering is $-\infty$ (∞). The floating-point intervals are the subset of the real intervals where a bound, if it exists, is a floating-point number. We assume that there are at least two finite floating-point numbers. As a result, the empty subset of $\mathcal{R}$ is also a floating-point interval.

The floating-point intervals have the property that for every set of reals there is a unique least floating-point interval that contains it. This property can be expressed by means of the function $\Box$ so that $\Box S$ is the smallest floating-point interval containing $S \subset \mathcal{R}$. Given a real number x, we denote by x^- the greatest floating-point number not greater than x, and by x^+ the least floating-point number not less than x.

By themselves, set extensions are not enough to obtain interval extensions. They need to be used in conjunction with the function $\Box$, as in the following definition of interval addition:

$$X + Y = \Box\{z \in \mathcal{R} \mid \exists x \in X, y \in Y . x + y = z\} \quad (2)$$

for all floating-point intervals X and Y. Compared with a definition such as

$$[a, b] + [c, d] = [(a + c)^-, (b + d)^+], \quad (3)$$

(which is equivalent for bounded intervals) (2) has the advantage of being applicable to unbounded intervals without having to define arithmetic operations between real numbers and entities that are not real numbers. Moreover, (2) includes the required outward rounding.

Similarly to (2) we have

Definition 6

$$X + Y \stackrel{def}{=} \Box\{z \in \mathcal{R} \mid \exists x \in X, y \in Y . x + y = z\}$$

$$
\begin{aligned}
X - Y &\stackrel{def}{=} \Box\{z \in \mathcal{R} \mid \exists x \in X, y \in Y . z + y = x\} \\
X * Y &\stackrel{def}{=} \Box\{z \in \mathcal{R} \mid \exists x \in X, y \in Y . x * y = z\} \\
X/Y &\stackrel{def}{=} \Box\{z \in \mathcal{R} \mid \exists x \in X, y \in Y . z * y = x\} \\
\sqrt{X} &\stackrel{def}{=} \Box\{y \in \mathcal{R} \mid \exists x \in X . y^2 = x\}
\end{aligned}
$$

Theorem 1 *The functions defined in Definition 6 map floating-point intervals to floating-point intervals, are defined for all argument floating-point intervals, and are set extensions of the corresponding functions from reals to reals.*

This is a summary of several results in [5].

Definition 7 *Let I be the set of intervals. $F \in I^n \to I$ is an* interval extension *of $f \in \mathcal{R}^n \to \mathcal{R}$ iff F is the restriction to domain $I^n \subset \mathcal{R}^n \to \mathcal{R}$ of a set extension of f. F is the* canonical interval extension *of f is defined to be $F(B) = \{f(x) \mid x \in B\}$ whenever this is an interval.*

4. Semantics of expressions via set theory

As all but a few exotic varieties of logic restrict functions to be total, we develop the semantics of expressions on the basis of set theory, even though most treatments of set theory also restrict functions to be total. However, as we have seen, functions in the usual set theory are easily generalized so that totality is not assumed. Modifying logic so that function symbols can be interpreted by partial functions has graver repercussions [2, 10].

Suppose that the expression e has the form $e_1 + e_2$ and that e_1 computes $f_1 : \mathcal{R}^m \to \mathcal{R}$ and that e_2 computes $f_2 : \mathcal{R}^n \to \mathcal{R}$. In such a situation, Jaulin et al. [6] (Theorem 2.2), suggest that the function f computed by e is the composition of $+$, f_1 and f_2.

But such a composition is not possible, as the types do not match, as required in Definition 3. We can make a composition if we form the Cartesian product of f_1 and f_2 and if we make additional assumptions about e_1 and e_2. To prepare these assumptions we need the following definition.

Definition 8 *Let $\{v_1, \ldots, v_n\}$ be the set of variables in expression e. The* variable sequence *of e is $\langle v_1, \ldots, v_n\rangle$ if the first occurrences of the variables in e are ordered according to this sequence.*

Consider the special case where $m = n$ and where e_1 and e_2 have the same variable sequence. Let $\delta \in \mathcal{R}^n \to \mathcal{R}^n \times \mathcal{R}^n$ with mapping

$$\langle x_1, \ldots, x_n\rangle \mapsto \langle\langle x_1, \ldots, x_n\rangle, \langle x_1, \ldots, x_n\rangle\rangle$$

As will be shown in Lemma 1, the function computed by $e_1 + e_2$ is $+ \circ (f_1 \times f_2) \circ \delta$. The types of δ, $f_1 \times f_2$, and $+$ do match: they are, respectively, $\mathcal{R}^n \to (\mathcal{R}^n \times \mathcal{R}^n)$, $(\mathcal{R}^n \times \mathcal{R}^n) \to \mathcal{R}^2$, and $\mathcal{R}^2 \to \mathcal{R}$. Thus it is clear the composition is defined and that its type is $\mathcal{R}^n \to \mathcal{R}$.

But it is of course a very special case if e_1 and e_2 have the same variables in the same order of first occurrence. To further illustrate what is needed to define a composition of $+$, e_1, and e_2, consider another special case: e_1 and e_2 have *no* variables in common, and their variable sequences are $\langle v_1, \ldots, v_m\rangle$ and $\langle w_1, \ldots, w_n\rangle$, respectively. As will be shown in Lemma 1, the function computed by $e_1 + e_2$ is again $+ \circ (f_1 \times f_2) \circ \delta$, except that δ is in $\mathcal{R}^{m+n} \to \mathcal{R}^m \times \mathcal{R}^n$ and has as map

$$\langle x_1, \ldots, x_m, y_1, \ldots, y_n\rangle \mapsto \langle\langle x_1, \ldots, x_m\rangle, \langle y_1, \ldots, y_n\rangle\rangle$$

Now the types of δ, $f_1 \times f_2$, and $+$, are, respectively, $\mathcal{R}^{m+n} \to (\mathcal{R}^m \times \mathcal{R}^n)$, $(\mathcal{R}^m \times \mathcal{R}^n) \to \mathcal{R}^2$, and $\mathcal{R}^2 \to \mathcal{R}$. Thus it is clear that the composition is defined and that its type is $\mathcal{R}^{m+n} \to \mathcal{R}$.

Finally, an example where the subexpressions share some, but not all variables. Consider the example where e_1 is $x * y$, e_2 is $y * z$, e is $e_1 + e_2$, and $\delta \in \mathcal{R}^3 \to (\mathcal{R}^2 \times \mathcal{R}^2)$ is such that δ maps as follows: $\langle x_1, x_2, x_3\rangle \mapsto \langle\langle x_1, x_2\rangle, \langle x_2, x_3\rangle\rangle$ for all $x_1, x_2, x_3 \in \mathcal{R}$. Now the functions f_1 and f_2 computed by e_1 and e_2 are the same function in $\mathcal{R}^2 \to \mathcal{R}$: it has as map $\langle s, t\rangle \mapsto s * t$ for all reals s and t. Yet the function computed by $e_1 + e_2$ does not have as map $s * t + s * t$: it is a different function, which is, however, described by the same formula $+ \circ (f_1 \times f_2) \circ \delta$.

These three examples suggest how to define in general, for any pair $\langle e_1, e_2\rangle$ of expressions and any domain D of interpretation, a "distribution function" that represents the pattern of co-occurrences of variables in e_1 and e_2.

Definition 9 *Given expressions e_1 and e_2 with variable sequences $\langle v_1, \ldots, v_m\rangle$ and $\langle w_1, \ldots, w_n\rangle$, respectively. Let D be a set of values suitable for substitution for the variables. Let $\{i_1, \ldots, i_p\}$ and let $\{j_1, \ldots, j_q\}$ be a partition in $\{1, \ldots, n\}$ such that $\{w_{i_1}, \ldots, w_{i_p}\}$ occur in e_1 and $\{w_{j_1}, \ldots, w_{j_q}\}$ do not occur in e_1*[1].

The distribution function *δ for the pair $\langle e_1, e_2\rangle$ and D is the function in $D^{m+q} \to D^m \times D^n$ that has as map*

$$\langle x_1, \ldots, x_m, y_{j_1}, \ldots, y_{j_q}\rangle \mapsto \langle\langle x_1, \ldots, x_m\rangle, \langle y_1, \ldots, y_n\rangle\rangle$$

for all $x_1, \ldots, x_m, y_1, \ldots, y_n$ in D.

[1] Hence the variable sequence of any expression of the form $e_1\langle$operation symbol$\rangle e_2$ is $\langle v_1, \ldots, v_m, w_{j_1}, \ldots, w_{j_q}\rangle$.

Definition 10 *An* interpretation *for an expression consists of a set D (the* domain *of the interpretation) and a map M that maps every n-ary operation symbol in the expression to a function in $D^n \to D$.*

A set extension I' of I is said to be continuous *if every symbol p is mapped to a continuous set extension of $M(p)$. I' is said to be* canonical *if every n-ary operation symbol p is mapped to a canonical set extension of $M(p)$.*

The distribution function specifies enough of the way variables are shared between two expressions to support the central definition of this paper:

Definition 11 *Let e be an expression and let I be an interpretation that maps each n-ary operation symbol in e to a function in $D^n \to D$, for $n \in \{1, 2\}$. We define by recursion on the structure of e, distinguishing three cases.*

Suppose e is a variable. The function computed by e under I *is the identity function on D.*

Suppose e is φe_1 where φ is a unary operation symbol. The function computed by e under I *is $f \circ f_1$, where f is the function in $D \to D$ that is the result of mapping by I of φ and where f_1 is the function computed by e_1 under I.*

Suppose e has the form $e_1 \diamond e_2$, where $\diamond$ is a binary operation symbol. Suppose δ is the distribution function for $\langle e_1, e_2\rangle$ and D. The function computed by e under I *is $\diamondsuit \circ (f_1 \times f_2) \circ \delta$, where $\diamondsuit$ is the result of mapping by I of $\diamond$.*

The definition assumes that no constants occur in expressions. We can simulate a constant by replacing it by a new variable and substituting the constant for that variable. In this way the definition does not suffer a loss of generality for expressions consisting of variables, constants, unary operators, and binary operators. At the expense of cumbersome notation (or sophisticated methods to avoid this), the function δ can be extended to cover n-ary operation symbols with $n > 2$.

The definition should conform to our intuition about expression evaluation. Suppose that D is the set of integers, that the functions computed by e_1 and e_2 yield 2 and 3, respectively. Then the definition should ensure that the function computed by $e_1 + e_2$ yields 5 when the interpretation maps $+$ to addition over the integers. The following lemma confirms this intuition in general for arbitrary binary operation symbols.

Lemma 1 *Let $e_1 \diamond e_2$ be the expression in Definition 11. Suppose that $\langle a_1, \ldots, a_m\rangle \in D^m$ is substituted for $\langle x_1, \ldots, x_m\rangle$ and that $\langle b_1, \ldots, b_n\rangle \in D^n$ is substituted for $\langle y_1, \ldots, y_n\rangle$. Let $\langle c_1, \ldots, c_q\rangle$ be such that $\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle$ is substituted for $\langle x_1, \ldots, x_m, y_1, \ldots, y_q\rangle$.*

It is the case that

$$f(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle) = f_1(\langle a_1, \ldots, a_m\rangle) \diamondsuit f_2(\langle b_1, \ldots, b_n\rangle),$$

where f is the function computed by $e_1 \diamond e_2$ according to Definition 11.

Proof:

$$\begin{aligned} f(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle) &= \\ (\diamondsuit \circ (f_1 \times f_2) \circ \delta)(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle) &= \\ (\diamondsuit \circ (f_1 \times f_2))\delta(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle)) &= \\ (\diamondsuit \circ (f_1 \times f_2))\langle\langle a_1, \ldots, a_m\rangle, \langle b_1, \ldots, b_n\rangle\rangle &= \\ \diamondsuit((f_1 \times f_2)(\langle\langle a_1, \ldots, a_m\rangle, \langle b_1, \ldots, b_n\rangle\rangle) &= \\ f_1(\langle a_1, \ldots, a_m\rangle) \diamondsuit f_2(\langle b_1, \ldots, b_n\rangle). & \end{aligned}$$

Lemma 2 *Let I be an interpretation for expression e and let I' be a set extension of I. Let f (f') be the function computed by e under the interpretation I (I'). Then f' is a set extension of f.*

Though a minor lemma in set theory, the special case where the domains of I and I' are the reals and intervals respectively, it plays the role of the Fundamental Theorem of Interval Arithmetic[2].

Proof: We proceed by induction on the depth of the expression. Suppose the lemma holds for all expressions of depth at most $n-1$. Let n be such that at least one of e_1 and e_2 is of depth $n-1$ and the other is of depth at most $n-1$. Suppose I has domain D and map M. Let e be $e_1 \diamond e_2$ and suppose that M maps $\diamond$ to $\diamondsuit$. Let δ be the distribution function of e_1 and e_2 in that order. Let f_1 and f_2 be the functions computed by e_1 and e_2, respectively, under I. Let f_1' and f_2' be the functions computed by e_1 and e_2, respectively, under I'. This gives as induction assumption that f_1' and f_2' are set extensions of f_1 and f_2.

Let f and f' be the functions computed from $e_1 \diamond e_2$ under interpretations I and I', respectively. Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be subsets of D containing the elements $a_1, \ldots, a_m, b_1, \ldots, b_n$. Let $c_1, \ldots, c_q$ be such that δ maps $\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle$ to $\langle\langle a_1, \ldots, a_m\rangle, \langle b_1, \ldots, b_n\rangle\rangle$.

Supposing that $\diamondsuit'$ is a set extension of $\diamondsuit$, we have

$$\begin{aligned} f(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle) &= \\ f_1(a_1, \ldots, a_m) \diamondsuit f_2(b_1, \ldots, b_n) &\in \\ f_1'(A_1, \ldots, A_m) \diamondsuit' f_2'(B_1, \ldots, B_n) &= \\ f'(A_1, \ldots, A_m, C_1, \ldots, C_q), & \end{aligned}$$

[2]Except that the statement in [4] inadvertently states instead the definition of interval extension.

which is the function computed by e under I'. Both equalities are justified by Lemma 1.

Theorem 2 *Let e be an expression with a variable sequence $\langle x_1, \ldots, x_n \rangle$. Let I be an interpretation for e, and I' a canonical set extension of I. Let f (f') be the function computed by e under the interpretation I (I'). If each variable x_i occurs only once in e, then f' is the canonical set extension of f.*

Proof: Following the same steps and notation as in the previous proof, we have

$$f'(A_1, \ldots, A_m, B_1, \ldots, B_n) =$$

(by Lemma 1)

$$f'_1(A_1, \ldots, A_m) \diamond' f'_2(B_1, \ldots, B_n) =$$

(by the induction assumption)

$$f_1(A_1, \ldots, A_m) \diamond' f_2(B_1, \ldots, B_n) =$$

(using that f'_1 and f'_2 are canonical set extensions and that e_1 and e_2 have no variables in common)

$$\begin{aligned} &\{y \in D \mid \exists y_1 \in f_1(A_1, \ldots, A_m), \\ &\exists y_2 \in f_2(B_1, \ldots, B_n). y = y_1 \diamond y_2\} \quad = \\ &\{y \in D \mid \exists a_1 \in A_1, \ldots, \exists a_m \in A_m, \\ &\qquad \exists b_1 \in B_1, \ldots, \exists b_n \in B_n. \\ &y = f_1(a_1, \ldots, a_m) \diamond f_2(b_1, \ldots, b_n) = \\ &\qquad f(a_1, \ldots, a_m, b_1, \ldots, b_n)\} \quad = \\ &\qquad f(A_1, \ldots, A_m, B_1, \ldots, B_n). \end{aligned}$$

5. Continuous set extensions

A fundamental fact in interval analysis can be stated intuitively as

> We can get arbitrarily close to the range of the point evaluation of an expression e by computing the interval evaluation of e with a sufficiently narrow interval.

So far we were only concerned with interval *arithmetic*. This fact, being a continuity property, gets us into the realm of analysis. So it is here that interval *analysis* begins.

As the validity of the statement and proof of such a property depends on a rigorous definition of the function computed by an expression, it is wise to revisit the concepts and the theorems.

Definition 12 *Let $\mathcal{F}$ be a family of sets of D. A sequence $S = \langle S_n \rangle_{n \in \mathcal{N}}$ of subsets of D* converges with respect to *$\mathcal{F}$ if it is nested, belongs to $\mathcal{F}$, and satisfies $\bigcap_{n \in \mathcal{N}} S_n = \{a\}$, where a is an element of D. We say that the singleton set $\{a\}$ is the* limit *of S.*

Definition 13 *Let $F \in \mathcal{P}(S) \to \mathcal{P}(T)$, and let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two families of sets of S and T, respectively.*

Let $A = \langle A_n \rangle_{n \in \mathcal{N}}$ be any convergent sequence w.r.t. $\mathcal{F}_1$ with limit $\{a\}$. F is continuous w.r.t. *$\mathcal{F}_1$ and F_2 in $\{a\}$ iff $\langle F(A_n) \rangle_{n \in \mathcal{N}}$ is a convergent sequence w.r.t. $\mathcal{F}_2$.*

Continuity is a very strong requirement. This raises the concern that no interesting examples might exist. The next lemma shows that this concern is unnecessary.

Definition 14 *Let $f \in \mathcal{R}^n \to \mathcal{R}$, and let $\|.\|$ be the Euclidean norm on $\mathcal{R}^n$. The function f is* Cauchy-continuous *at $c \in \mathcal{R}^n$ iff for every $\epsilon > 0$ there exists a $\delta > 0$ such that $\|x - c\| \leq \delta$ and $x \in dom(f)$ imply that $|f(x) - f(c)| \leq \epsilon$.*

A sequence $\langle x_i \rangle_{i \in \mathcal{N}}$ with $x_i \in \mathcal{R}^n$ for all $i \in \mathcal{N}$ is Cauchy-convergent *to $\xi \in \mathcal{R}^n$ iff for every $\epsilon > 0$ there exists an n such that $\|\xi - x_i\| \leq \epsilon$ for all $i > n$.*

Lemma 3 *Let $f \in \mathcal{R}^n \to \mathcal{R}$ be Cauchy-continuous at every $x \in dom(f)$ and suppose f has a canonical interval extension F. Then F is continuous w.r.t. the family of boxes of $\mathcal{R}^n$, and the family of intervals of $\mathcal{R}$.*

Proof: Suppose that x is an element of $\mathcal{R}^n$, and that $\langle B_n \rangle_{n \in \mathcal{N}}$ is a sequence of boxes in $\mathcal{I}^n$ that converges to x w.r.t. the family of boxes of $\mathcal{R}^n$. To prove that F is continuous w.r.t. the family of boxes of $\mathcal{R}^n$, and the family of intervals of $\mathcal{R}$, we have to show that the sequence $\langle F(B_n) \rangle_{n \in \mathcal{N}}$converges w.r.t. the family of intervals of $\mathcal{R}$. It is clear that this sequence is nested and belongs to the family of intervals of $\mathcal{R}$. So, we only need to show that $\bigcap_{n \in \mathcal{N}} F(B_n)$ is a singleton. In fact,

$$\bigcap_{n \in \mathcal{N}} F(B_n) = \{f(x)\}.$$

The following inclusion is obvious: $\{f(x)\} \subset \bigcap_{n \in \mathcal{N}} F(B_n)$. Let y be an element of $\bigcap_{n \in \mathcal{N}} F(B_n)$. This implies that for every $n \in \mathcal{N}$, there exists x_n in B_n such that $f(x_n) = y$. Because $(B_n)_{n \in \mathcal{N}}$ is a nested sequence of boxes that intersect in $\{x\}$, the sequence $(x_n)_{n \in \mathcal{N}}$ Cauchy-converges to x. Since f is Cauchy-continuous at x, we have $f(x) = y$. Therefore, $\bigcap_{n \in \mathcal{N}} F(B_n) \subset \{f(x)\}$, which proves the lemma.

Lemma 4 *Let $f \in S \to T$, and let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two families of sets of S and T, respectively. Let F be a continuous set extension of f w.r.t. $\mathcal{F}_1$ and $\mathcal{F}_2$ and let $A = \langle A_n \rangle_{n \in \mathcal{N}}$ be a convergent sequence w.r.t. $\mathcal{F}_1$ with limit $\{a\}$. Then $F(\{a\}) = \{f(a)\}$.*

Proof: As F is continuous w.r.t. $\mathcal{F}_1$ and $\mathcal{F}_2$, $\langle F(A_n) \rangle_{n \in \mathcal{N}}$ is a convergent sequence w.r.t. $\mathcal{F}_2$ with limit, say, $\{b\}$. As F is a set extension of f we have

that $\{f(x) \mid x \in A_i\} \subset F(A_i)$, for all $i \in \mathcal{N}$. As $a \in A_i$ for all $i \in \mathcal{N}$, we have that $f(a) \in \{f(x) \mid x \in A_i\}$ for all $i \in \mathcal{N}$. Hence $f(a) \in \bigcap_{i\in\mathcal{N}} F(A_i) = \{b\}$. So we must have $f(a) = b$.

We are interested in interval extensions that are not canonical, yet are continuous.

Starting from a family $\mathcal{F}$ of sets of a set D, we can construct a family of sets $\mathcal{F}_n$ of D^n, for any natural number n, by taking all the Cartesian products of any n sets in $\mathcal{F}$. So, for any natural number n, and for any function $F \in \mathcal{P}(D^n) \to \mathcal{P}(D)$, we can study the continuity of F w.r.t. $\mathcal{F}_n$ that was constructed from $\mathcal{F}$. In this way, we treat the continuity of F by referring to $\mathcal{F}$ instead of $\mathcal{F}_n$.

In what follows, we suppose that the family of sets $\mathcal{F}$ of the domain D of an interpretation is given, and that the continuity of a set extension of an n-ary operation is based on this family. So, we will not use "w.r.t." from now on. In the case where D is $\mathcal{R}$, $\mathcal{F}$ is the family of intervals in $\mathcal{R}$.

Definition 15 *Let I be an interpretation with domain D and map M. A set extension I' of I is said to be* continuous *if every symbol p is mapped to a continuous set extension of $M(p)$. I' is said to be a* canonical interval extension *of I iff every symbol p is mapped to a canonical interval extension of $M(p)$.*

Theorem 3 *Let e be an expression. Let I be an interpretation for e, and let I' be a continuous set extension of I. Let f (F) be the function computed by e under the interpretation I (I'). Then F is a continuous set extension of f.*

Proof: From Lemma 2, the function F is a set extension of f. So we only need to prove that F is continuous. To do so, we proceed by induction on the depth of the expression e. The theorem holds when e has no subexpressions, that is, when e is a variable. In that case f and F are the identity functions, independently of I and I'. The identity function in $\mathcal{P}(D) \to \mathcal{P}(D)$ is continuous.

This takes care of the base of the inductive proof. Let the induction assumption be that the theorem holds for all expressions of depth at most $d-1$. Let e be the expression $e_1 \diamond e_2$, where one of the subexpressions has depth $d-1$ and the other has depth at most $d-1$. Suppose that the interpretation I has domain D and maps $\diamond$ to $\diamondsuit$. Let the interpretation I' have $\mathcal{P}(D)$ as domain and let it map $\diamond$ to $\diamondsuit'$, a continuous set extension of $\diamondsuit$. Let δ be the distribution function with D for e_1 and e_2 in that order. Let $c_1, \ldots, c_q$ be such that δ maps $\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle$ to $\langle\langle a_1, \ldots, a_m\rangle, \langle b_1, \ldots, b_n\rangle\rangle$.

Let F, F_1, and F_2 be the functions computed under I' by e, e_1, and e_2, respectively. Suppose that $\langle A_1^i\rangle_{i\in\mathcal{N}}, \ldots, \langle A_m^i\rangle_{i\in\mathcal{N}}$ and $\langle B_1^i\rangle_{i\in\mathcal{N}}, \ldots, B_n^i\rangle_{i\in\mathcal{N}}$ are sequences of subsets of D that converge respectively to $\{a_1\}, \ldots, \{a_m\}$ and $\{b_1\}, \ldots, \{b_n\}$. According to the induction assumption F_1 and F_2 are continuous set extensions. This implies that $\langle\langle F_1(A_1^i, \ldots, A_m^i), F_2(B_1^i, \ldots, B_n^i)\rangle\rangle_{i\in\mathcal{N}}$ converges to $\{\langle f_1(a_1, \ldots, a_m), f_2(b_1, \ldots, b_n)\rangle\}$, by Lemma 4.

Let $\langle C^i\rangle_{i\in\mathcal{N}}$ be any such that $\delta(C^i) = \langle\langle A_1^i, \ldots, A_m^i\rangle, \langle B_1^i, \ldots, B_n^i\rangle\rangle$ and such that $\langle C^i\rangle_{i\in\mathcal{N}}$ converges to $\{\langle\langle a_1, \ldots, a_m\rangle, \langle b_1, \ldots, b_n\rangle\rangle\}$

We show that F is a continuous set extension of f by showing that $\langle F(C^i)\rangle_{i\in\mathcal{N}}$ converges to $\{f(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle)\}$. To do so, we need to show that the sequence $\langle F(C^i)\rangle_{i\in\mathcal{N}}$ is nested, and that $\bigcap_{i\in\mathcal{N}} F(C_i)$ is the right value, namely $\{f(\langle a_1, \ldots, a_m, c_1, \ldots, c_q\rangle)\}$.

$$F(C^{i+1}) =$$

(by Definition 11)

$$(\diamondsuit' \circ (F_1 \times F_2) \circ \delta)(C^{i+1}) =$$

(by application of δ)

$$(\diamondsuit' \circ (F_1 \times F_2))(\langle\langle A_1^{i+1}, \ldots, A_m^{i+1}\rangle, \langle B_1^{i+1}, \ldots, B_n^{i+1}\rangle\rangle) =$$

(by Definition 4)

$$\diamondsuit'(\langle F_1(\langle A_1^{i+1}, \ldots, A_m^{i+1}\rangle), F_2(\langle B_1^{i+1}, \ldots, B_m^{i+1}\rangle)\rangle) \subset$$

(by the induction assumption and continuity of $\diamondsuit'$)

$$\diamondsuit'(\langle F_1(\langle A_1^i, \ldots, A_m^i\rangle), F_2(\langle B_1^i, \ldots, B_m^i\rangle)\rangle) = F(C^i),$$

which proves that $\langle F(C^i)\rangle_{i\in\mathcal{N}}$ is nested. As for the convergence to the right value, we observe the following:

$$\bigcap_{i\in\mathcal{N}} F(C^i) =$$

(by Definition 11)

$$\bigcap_{i\in\mathcal{N}} (\diamondsuit' \circ (F_1 \times F_2) \circ \delta)(C^i) =$$

(by application of δ)

$$\bigcap_{i\in\mathcal{N}} (\diamondsuit' \circ (F_1 \times F_2))(\langle\langle A_1^i, \ldots, A_m^i\rangle, \langle B_1^i, \ldots, B_n^i\rangle\rangle) =$$

(by Definition 4)

$$\bigcap_{i\in\mathcal{N}} \diamondsuit'(\langle F_1(\langle A_1^i, \ldots, A_m^i\rangle), F_2(\langle B_1^i, \ldots, B_m^i\rangle)\rangle) =$$

(by continuity of $\diamond'$)

$$\diamond'(\langle \bigcap_{i\in\mathcal{N}} F_1(\langle A_1^i,\ldots,A_m^i\rangle),\ \bigcap_{i\in\mathcal{N}} F_2(\langle B_1^i,\ldots,B_m^i\rangle)\rangle) =$$

(by the induction assumption)

$$\diamond'(\langle\{f_1(\langle a_1,\ldots,a_m\rangle)\},\{f_2(\langle b_1,\ldots,b_n\rangle)\}\rangle) =$$

(by Lemma 4)

$$\{f_1(\langle a_1,\ldots,a_m\rangle)\}\diamond\{f_2(\langle b_1,\ldots,b_n\rangle)\} =$$

(because f is the function computed by $e_1 \diamond e_2$)

$$\{f(\langle a_1,\ldots,a_m,c_1,\ldots,c_q\rangle)\},$$

which shows that $F = \diamond' \circ (F_1 \times F_2) \circ \delta$ is a continuous set extension of f, the function computed by e.

Corollary 1 *Let $f \in \mathcal{R}^n \to \mathcal{R}$ be the function computed by an expression e under an interpretation I that assigns Cauchy-continuous functions to the operation symbols in e. Let F be the function computed by e under the canonical interval extension of I. Let $\langle A_i\rangle_{i\in\mathcal{N}}$ be nested boxes converging to $\{a\}$. Then $\langle F(A_i)\rangle_{i\in\mathcal{N}}$ is a sequence of nested intervals converging to $\{f(a)\}$.*

In interval analysis, this corollary plays the role of Fundamental Theorem.

Proof: Since the image of any box by a Cauchy-continuous function is an interval, the interval extension associated with each operation symbol is canonical (every Cauchy-continuous function has a canonical interval extension). Using Lemma 3, these interval extensions are continuous. By Theorem 3, F is continuous. By Definition 13, $\langle F(A_i)\rangle_{i\in\mathcal{N}}$ converges to $\{f(a)\}$.

6 Conclusions

The fact that the result of an expression evaluation in intervals gives a result that contains the range of values of the function computed by the expression cannot be a mathematical theorem without a mathematical definition of what it means for a function to be computed by an expression. In this paper we give such a definition and prove the theorem on the basis of it.

Another fundamental assumption in the use of intervals is that, as we make the intervals in an interval evaluation of an expression narrower, the interval result gets closer to the range of values of the function computed by the expression. We use our definition to prove a theorem to this effect.

Our starting point in all this is that intervals are sets and that, therefore, interval extensions of functions are set extensions of functions. The latter concept is an old one in set theory and is more widely applicable. Our definition and two main theorems are stated in terms of sets, so apply to intervals as special cases.

This is of course only of interest to those who believe in sets as foundation of mathematics. A radically different approach to the fundamental theorems of interval analysis is found in Paul Taylor's work (see for example [13]). Here the starting point is topology, axiomatically founded rather than set-theoretically.

If it seems that our proposed foundations for interval methods are overly complex in comparison with the way they are given in the literature, we are comforted by Einstein's dictum: *Make things as simple as possible, but not simpler.*

7 Acknowledgements

This research was supported by the University of Victoria and by the Natural Science and Engineering Research Council of Canada. We owe a great debt of gratitude to our anonymous reviewer whose extremely detailed and helpful report has helped us to improve this paper.

References

[1] N. Bourbaki. *Théorie des Ensembles (Fascicule de Résultats)*. Hermann et Cie, 1939.

[2] S. H. Cheng and C. B. Jones. On the usability of logics which handle partial functions. In *Proc. of the 3rd. Refinement Workshop*, pages 51–69, 1991.

[3] P. R. Halmos. *Naive Set Theory*. D. Van Nostrand, 1960.

[4] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, 1992.

[5] T. Hickey, Q. Ju, and M. van Emden. Interval arithmetic: from principles to implementation. *Journal of the ACM*, 48(5):1038 – 1068. 2001.

[6] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer-Verlag, 2001.

[7] J. L. Kelley. *Topology*. D. Van Nostrand, 1955.

[8] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[9] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.

[10] D. L. Parnas. Predicate logic for software engineering. *IEEE Trans. Softw. Eng.*, 19(9):856–862, 1993.

[11] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood/John Wiley, 1984.

[12] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Ellis Horwood/John Wiley, 1988.

[13] P. Taylor. Interval analysis without intervals (extended abstract). In G. Hanrot and P. Zimmermann, editors, *Real Numbers and Computers 7*, pages 41–45, 2006. Nancy.

Expression Defined Accuracy

A. Pokorny
Elektrobit Automotive
D 91058 Erlangen
andreas.pokorny@elektrobit.com

J.Wolff von Gudenberg
University of Würzburg
Department of Informatics
Am Hubland
D 97074 Würzburg
wolff@informatik.uni-wuerzburg.de

Abstract

In numerical computations the accuracy of the result quite often depends on a few expressions. In numerical linear algebra, e.g., summations or dot products should very often be computed with additional precision or accuracy. Corresponding algorithms have been developed for a long time and only recently revisited. The usage of these algorithms would be facilitated, if we had a means in a programming language to specify the accuracy requirements of an expression evaluation.

In this paper we present a precision aware C++ template library (PTL) for matrix / vector operations that provides several algorithms with different accuracy or precision characteristics for matrix multiplication and related operations. A matrix is a template parameterized with the number of rows and columns, the element type, a type representing the shape, and an evaluation strategy. Currently only two shapes are implemented, fixed or dynamically adaptable dense arrays. We distinguish between row and column vectors. The access to submatrices and -vectors is accomplished by an overloaded function template. It is possible to adapt the expression system to types declared in other libraries or declared by the user.

The concept of expression templates is extended in a way that allows the user to specify rules for the evaluation strategy. The expression tree is constructed by overloading the operators for the expression type. In a second but still compile-time step the evaluation strategy is chosen and the trees are transformed and prepared for run-time execution. The strategy is determined by the tag type of the result, but it can be explicitly set using the index operator.

The evaluation strategies can be combined with loop unrolling or loop fusion. Note that the latter not only increases the precision but also the accuracy of the result, since this strategy directly implements the dotprecision expression evaluation in the XSC languages.

The library provides evaluation strategies for matrix and vector expressions with k-fold precision and with least bit accuracy. Efficiency and accuracy of the algorithms are tested vs. the Gnu multiple precision library GMP.

1. Introduction

An efficient, reliable, accurate library for matrix or vector arithmetic is the backbone of many scientific codes. The design and implementation of such libraries, hence, is an everlasting topic in research and development of scientific algorithms. In this paper we present a library for linear algebra that allows to evaluate expressions with varying precision or accuracy. We use the term precision to specify the number of bits used in an operation on input data and accuracy to characterize the number of correct bits of the result. Note that these may be different in expressions. If the expression $a + b - a$ is evaluated with double precision (53 bit) for $a = 1e50$ and $b = 1$, e.g., the result is 0 with no bit of accuracy. That simple example may motivate the quest for more accurate expression evaluation algorithms which indeed have been developed a long time ago [5]. A key component of those algorithms often is the computation of accurate sums or dot products which have quite recently been revisited in two papers [12, 11].

In our precision aware template library PTL we use the concept of expression templates [14] to evaluate linear algebra operations. We separate expression evaluation into 4 steps: In the first step the expression tree is built, then the evaluation strategy is determined, thirdly the expression is transformed according to the strategy, and finally evaluated. Rules are used to check for the validity of the generated expression. These rules can be extended by the user. Usually the strategy is given by the resulting object, but it can be freely chosen, and new strategies may be added. Transformation rules for standard strategies and data types are provided by the library. The user can specify rules for adaption

of foreign types. After the appropriate transformation the evaluation is straightforward.

In the following section 2 we give more details about this 4-step expression evaluation, how it is implemented in our PTL. A discussion and comparison with other libraries is performed in section 3.

2. Rule Based Expression Templates for Linear Algebra

2.1. Overview

The library provides data type templates for matrices and vectors parameterized with value type, shape and dimension. In the frontend operators and functions for these types are overloaded to obtain a basic expression type. The expressions are transformed and evaluated according to various strategies by the backend. (see Figure 1)

2.2. Concepts of Data Types

In our precision aware linear algebra library (PTL) expressions can be evaluated via different strategies in order to fulfill the increased accuracy or precision requirements we have in mind. Hence data types, i.e. vectors and matrices, are parameterized with their element type that usually is a scalar type, a fixed or dynamic dimension type, the shape of storage allocation which currently means densely packed rectangular structures, and the strategy type for evaluation.

This leads to the definition of the matrix concept where we use the proposed N2081 syntax[4].

```
concept matrix < typename value_type > {
  typename result_type;
  typename dimension_type;
  typename storage_type;
  typedef matrix_tag result_tag;
  typename strategy_type;

  result_type operator()
         (size_t row, size_t column ) const;
  result_type & operator()
         (size_t row, size_t column );

  dimension_type dimension() const;

}
```

Listing 1. matrix concept

The strategy type and the second () operator are optional.

We model row vectors and column vectors as different concepts, each with static or dynamic dimension, hence 4 similar templates exist.

For the element type (`value_type`) arithmetic operators and a parameterless constructor are required. Hence the standard floating-point types as well as appropriate user defined types can be used.

2.3. Expression Handling

Currently we provide the standard operators for matrix-vector arithmetic with different strategies available for matrix and matrix-vector multiplication. Since we have separate types for row and column vectors, we have 14 different overloadings of the multiplication operator. Some of these mean dot products and deserve a specific handling, others, like multiplication with a scalar value, do not.

We need a delayed and flexible evaluation of expressions, therefore we proceed in four steps:

1. The expression is parsed and checked for validity. If it is invalid an appropriate error message has to be produced.

2. The evaluation strategy is taken from the target object, either implicitly or by direct call of the index operator.

3. The expression is transformed according to the strategy.

4. Code for the evaluation strategy is generated.

A general overview of our treatment of expressions is given in Figure 1.

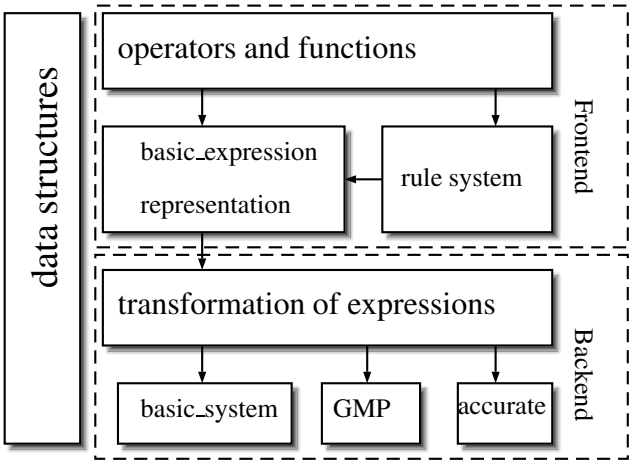


Figure 1. Treatment of Expressions

2.4. Expression Encoding

An expression consisting out of binary or unary operations is parsed into a syntax tree where inner nodes are operators and the leaves are the operands. The subtrees are expressions as well, either unary or binary. Because their structure is known, they can be obtained by inheritance from the general expression type constrained by an instantiation with themselves. This is known as the curiously recurring template pattern [3].

The `basic_expression` template together with its substructures for binary or unary expressions realizes the expression type. The substructures are parameterized by the operand type(s), the dimension type as well as the result and operator tag. The operators are overloaded so that each application of an operator generates an expression tree.

2.5. Rules

The result type of an overloaded operator is a function of the operand types. Validity of the instantiation has to be checked as well as appropriate actions or transformations have to be performed. Usually the conditions are checked by each operator. But since there are similar or even identical constraints for different operators, we separate the rules for binary or unary operators into two meta functions that are applied at compile time to the result type. (A meta function is a template and application means partial instantiation.) These rules refer to the language rule "Substitution failure is not an error" (SFINAE), i.e. instantiations that would lead to illegal types are not performed by the compiler and thus do not provoke an error.

The SFINAE language feature is used by the boost enable_if utility [1] that provides a means to build a powerful compile time overload resolution system. We further intensively use the boost meta programming library [1].

A rule is a partial specialization of a template for unary or binary operators, respectively. That basic template meta function depends on the operand type or types, the operator type tag and a hidden type parameter with default `void`, it is derived from the false constant of the boost metaprogramming library (`mpl::false_`). Every single specialisation, i.e. each concrete rule should therefore inherit from `mpl::true_`, the true constant.

One rule splits all matrix and vector dot product multiplications in the input tree into 2 operations, the binary inner product and the unary accumulation or summation. Hence, this rule leads to different expression trees for the same source code depending on the data types, see figure 2. That gives way to define more accurate versions of dot products.

An other rule can be applied for all operator tags which require equal dimensions. In the body that defines the

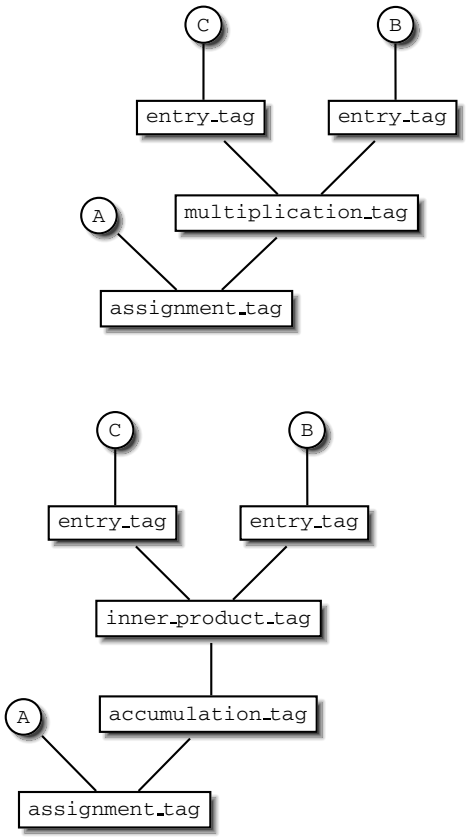


Figure 2. Two Expression Trees for $A = C * B$; up) C :=vector, B :=scalar bottom) C :=matrix, B :=vector

`result_type` the corresponding runtime checks, using assertions, are performed, if at least one dimension is dynamic. Several other meta functions are called to assemble all aspects of the actually encoded expression type.

In each rule a part of the expression tree is generated. The `handle_value_tag` meta function ensures that values are wrapped into unary expressions in order to match our expression representation. Sometimes the tree is slightly reorganised. Combined assignment operators like "+=", e.g. are split into operation and assignment to increase the flexibility of the subsequent evaluation. In the default case they are reunited by the compiler optimization.

So far the rule framework can handle only types that inherit from one of the basic expression types. The `adapt_type` template enables the insertion of literals, primitive variables or user defined types into the expression tree as left or right hand operands. That meta function is structured like the rule templates, i.e. it defines a `result_type` that is returned by an init function. It has to be instantiated with a foreign type and the operator tag in an operator or function definition. Additional wrappers and

adapter functions are necessary, if a foreign type is used as target of an assignment, since the assignment operator cannot be globally overloaded.

The rule framework has factored out the tedious compatibility tests at many places. The rules only depend on the properties of an expression how they are visible as template parameters of basic expressions. The genuine semantics of the operators in a matrix vector library is not relevant at this point.

2.6. Expression Transformation and Evaluation

Expressions are transformed and evaluated with respect to a given strategy. The expression tree, a strategy type and a target object for the results are specified, and then the eval function proceeds as follows

- append assignment node, if needed
- transform input tree
- optionally generate temporary data structure
- evaluate
- optionally assign temporary data structure

The expression tree is transformed into a tree out of basic nodes that may be unary or binary nodes, respectively. The first template parameter of a node is the strategy tag. A strategy is thus defined as a collection of partially specialized binary or unary node templates, that use the tag type of the strategy as first parameter. Each valid combination of operand types and operator tag matches to a particular structure tempalte. Usually a new strategy extends the default single strategy `basic_system`. In the default strategy the evaluation of a unary node with the accumulation tag set generates a simple for loop to sum all entries of a vector. The strategy `kfold_dot_product` in turn calls a more sophisticated algorithm, see section 2.7.

Because these two strategies only refer to one node in the expression tree, they could have been called on the original tree. But since some strategies may need a differently structured tree, we provide an explicit `transformer` template with a given strategy and an expression tree type as template arguments. That template calls the meta functions `get_unary_node`, `get_binary_node` to construct the new tree. These inherit from `get_node`, a template whose two arguments are a strategy and a meta function class.

Inside `get_node` we check for a single strategy, if the instantiation is defined, and delegate the call to the parent strategy, if not. The default implementation for the `unary_node` and `binary_node` template is derived from a structure called `undefined`. The partial specializations of these node templates that belong to a strategy, are not derived from that structure. Hence the check we perform for a strategy is implemented with the derivation test `is_base_and_derived` of Boost.TypeTraits.

Note that sequences of strategies can be stored as type lists using an overloaded comma operator. In this case the meta function `get_node` has to test all strategies of the sequence, as described above. If there is no partial specialization for any of the tested strategies, the whole sequence is transformed with a meta function that returns the base strategy, if there is any.

Depending on the implemented operator tag, the node class templates of the default strategy provide different methods for evaluation. The assignment nodes, which are always at the root of the tree provide an execute method. For other operator tags paranthesis operators, are implemented, that provide access to the data or the result of the evaluation of the respective subtree.

2.7. Strategies for Dot Products

The standard strategy transforms a dot product multiplication into two operations, the componentwise inner product and the accumulation, and then calculates these 2 operations with standard precision. As dot products often occur in expressions that are crucial for the overall accuracy of the result various more precise or even more accurate algorithms have been developed in the past.

The simplest modification is to compute each operation with doubled or k-fold precision. This can be simulated without changing the data format by error free floating-point multiplication and summation. These operations which deliver a result and a remainder term as output are well-known for a long time [10] and have been reconsidered in [12].

Let us treat the function `two_sum` for error free addition in more detail.

```
template<typename T>
void two_sum( T a, T b, T& sum, T& rem) {
  sum = a + b;
  T z = sum - a;
  rem = (a - (sum - z)) + (b - z);
}
```

We then have $a+b = sum+rem$ and mantissae of sum and rem do not overlap. Since in real arithmetic $sum = a + b$ and $rem = 0$, we have to stop the compiler from aggressive optimization in this function.

In a similar way we can define error free multiplication that uses a splitting aof the factors into two half-long floating-point numbers.

For the accumulation of a vector (p_i) the error free addition can be cascaded.

```
s[1] = p[1];
for(i=2;i<=n;i++) {
   two_sum(s[i-1], p[i], s[i], q[i]);
}
```

After this first path, we have in real, unrounded arithmetic

$$s = \sum_{i=1}^{n} p_i = s_n + \sum_{i=2}^{n} q_i$$

Note that we have not lost any information up to this point, we only condensed the main part of the sum in s_n and kept all the remainders. The mantissae of s_n and q_n do not overlap. Hence, we can continue with the accumulation of s_n and the q_i.

The template function `sum_k_vert<int K>` performs this accumulation up to depth `K` thus producing a result in k-fold precision. For a dotproduct the vector is prepared from the values and remainders of the error free inner product operation.

These strategies enlarge the precision of the result, further iteration can be used to guarantee a specific accuracy. e.g. Bohlender already proved in [2] that maximal accuracy is achieved for $n-1$ iterations. In [11] these algorithms are revisited and many more variants are proposed. We implemented one of those as our strategy `accurate_system`.

2.8. Instantiation of Expressions with Dotproducts

The main message of the paper is to show that instantiation of expression templates according to different strategies is possible. We, therefore, illustrate the treatment of

`y[kfold_dot_product<K>]=A*x`

in more detail. Consider the following program fragment [1]:

```
int main ()
{
  ptl::fixed_matrix<float,10,10> A(0.0f);
  A = init_mat();
  ptl::fixed_cvector<float,10> x(0.0f);
  x = init_col();

  y[kfold_dot_product<4>] = A*x;
}
```

Listing 2. Matrix vector multiplication

The matrix and column vector templates are instantiated as unary expressions for which the $*$ operator is overloaded to obtain the following expression.

```
binary_expression<
  unary_expression<
    fixed_matrix<float,10,10>
    , fixed_dimension<10,10>
    , matrix_tag
    , value_tag>
  , unary_expression<
    fixed_cvector<float,10>
    , fixed_dimension<10,1>
    , column_vector_tag
    , value_tag>
  , fixed::dimension<10,1>
  , column_vector_tag
  , multiplication_tag>
```

Listing 3. Expression tree

At this stage the rule for scalar products fires and rearranges the expression tree.

```
template<class LT,class RT,class OpT>
struct rule_2<LT,RT,OpT,
  typename boost::enable_if<
    boost::mpl::and_<
      is_same<mul_tag,OpT>
      , have_mul_dimensions<LT, RT>
      , have_mul_result_tags<LT,RT>
      , mul_incl_scalar_product<LT,RT>
      >
    >::type
  >
  : boost::mpl::true_
{
  typedef mul_dimension<LT, RT> dim_picker;
  typedef binary_expression<
    typename handle_value_tag<LT>::type
    , typename handle_value_tag<RT>::type
    , typename dim_picker::type
    , typename mul_result_tags<LT,RT>::type
    , inner_product_tag>
    inner_expression_type;
  typedef unary_expression<
    inner_expression_type
    , typename dim_picker::type
    , typename mul_result_tags<LT,RT>::type
    , accumulation_tag>
    result_type;
  static result_type init( LT l, RT r )
  {
      return result_type(
        inner_expression_type (
          handle_value_tag<LT>::handle(l)
          , handle_value_tag<RT>::handle(r)
          , dim_picker::init(l.dim(), r.dim())
          )
        , dim_picker::init(l.dim(), r.dim())
        );
  }
};
```

Listing 4. Rule transforms expression tree

[1] In the listings we have abbreviated the names, and left out some const modifiers

```
template<class Strategy, class T, class DT
  , class RTag, class OTag>
struct transformer<Strategy,
  unary_expression<T, DT, RTag, OTag>,void>
{
  typedef transformer<Strategy,T> transed;

  typedef typename get_unary_node<
    Strategy
    , typename transed::type
    , DT
    , RTag
    , OTag
    >::type type;

  static type init(
   unary_expression<T,DT,RTag,OTag> expr){
    return type( expr.dimension(),
        transformed::init( expr.operand ));
  }
};
```

Listing 5. Backend transformation of the tree

The `transformer` meta function transforms the expression tree into a tree of nodes that can be evaluated according to the given strategy. We only show the code for unary expressions.

The left hand side of the assignment defines the evaluation strategy that influences the transformation of the tree. The following node structure template will be instantiated during the transformation process.

```
template<size_t K,class Indexer,
    class DimT,class ResultTag,class ET>
struct unary_node<kfold_dot_product<K>,
Indexer,DimT,ResultTag,accumulation_tag,ET>
    : basic_node<DimT,ResultTag
     ,accumulation_tag>,
     defined
{
...
  void prepare_values( CT cursor, CT end ) {
  ...
    for( size_t i = 1; cursor != end;
        ++cursor, ++i ) {
      value_type h;

      two_product(
        indexer.left_value(cursor.left)
        , indexer.right_value(cursor.right)
        , h
        , values[i] );

      two_sum( acc, h,
          acc, values[offset+i-1]);
```

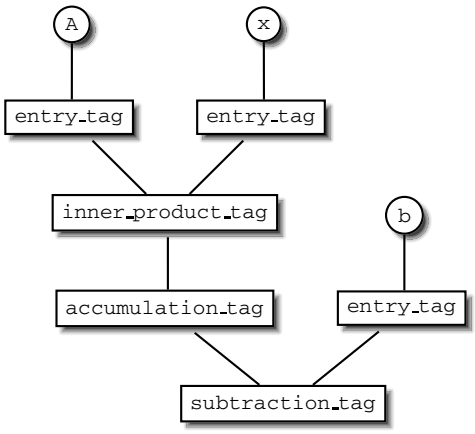


Figure 3. Strategy dotproduct evaluation

```
    }
    values.back() = acc;
  }

  result_type operator()(size_t row,
                         size_t column) {
    prepare_values( cursor, end );
    return sum_k_vert<K-1>(values);
  }
};
```

Listing 6. Evaluation of expression tree

When the tree is assigned or access by the $()$ operator, code is generated for the proper strategy.

2.9. Dotproduct Expressions

Dotproduct expressions [15], i.e. dot products built with more than one operator, can be evaluated with arbitrary accuracy, if the summation is defered further. They frequently occur in error correction methods where high accuracy is essential. For this purpose we define a new strategy `dotexpression` that further transforms the expression tree.

The difference is illustrated for the defect term $d = Ax - b$ in Figure 3 and Figure 4. We recall the semantics of our `accumulation_tag` is that all arguments are collected and summed up according to the strategy.

For Figure 4 i.e.for the new strategy `dotexpression` this means that the exact sum with one final rounding is computed.

In Figure 3 on the other hand the final subtraction suffers from high cancellation, even if the accumulation of the matrix-vector product has been performed with full accuracy.

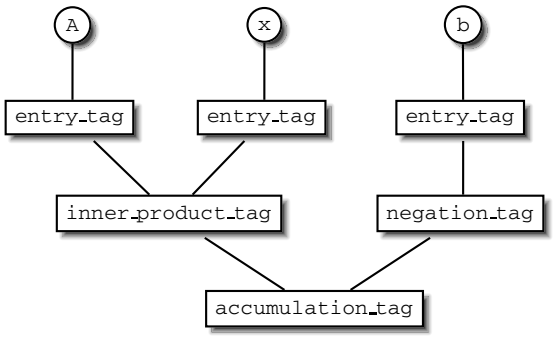


Figure 4. Strategy dotexpression evaluation

2.10. Strategies for Efficiency

For efficiency we implemented the unrolling of loops, a block based strategy and two different SIMD strategies (3DNOW and SSE2) exploiting the new architectures and extensions of the instruction set for vector processing. First tests show a moderate acceleration for unrolling. The other strategies currently suffer from temporary objects and copying, so more sophisticated implementations will have to be developed.

These strategies compute the expression blockwise, without imposing special memory alignment and storage layout requirements on the participating data structures. Therefore blocks have to be filled elementwise prior to the calculation, at the leaves and in certain nodes inside the tree. A node of these strategies always returns blocks containing the values of adjacent results. This overhead was not optimized by the compiler, so depending on the chosen block size the resulting binary code was either slower or as fast as an elementwise evaluation without these instructions. We believe that a different evaluation approach could avoid the unnecessary load and store operations, but that topic was not investigated any further.

2.11. Validation and Test

To validate our implementation of the dot product strategies, we generated ill-conditioned dot products and compared the results with a GMP strategy that evaluates the expressions with arbitrary precision.

We also measured the time for various strategies. As expected the time for the accurate strategy depends on the condition number. The other algorithms do not produce trustable results.

Runtime of x*y with conditioned vectors (input type: double, length: 500, on AMD64 3000+ with GNU/Linux and g++-4.1.1 using GMP-4.2.1; all libraries and benchmarks were compiled with "-O3 -march=athlon64")

The figure displays the runtime of an accumulation for increasing condition number.

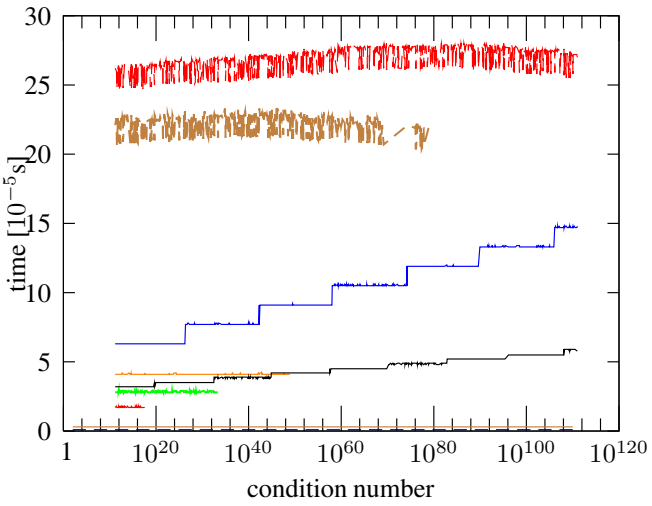


Figure 5. Conditioned Runtime

- Recursive sum, no expression templates(– –);
- default evaluation(——) ;
- kfold_dot_product< 2 >(——) ;
- kfold_dot_product< 3 >(——) ;
- Accurate Sum with double(——) ;
- kfold_dot_product< 4 >(——) ;
- Accurate Sum with long double(——) ;
- GMP 256 bit (– –) ;
- GMP 512 bit(– –) ;

All graphs but the recursive sum and default evaluation are restricted to runs satisfying: $e_{relative} \leq 1 \times 10^{-15}$

2.12. Extension of the Library

Although the architecture of the PTL library is quite involved, it is relatively easy to extend. New functions can be entered by

1. defining a new operator tag
2. specifying a rule for the correct use
3. defining two frontend functions one for foreign types and one for known types
4. extending a strategy by defining a new specialisation of a node structure

A new strategy can be introduced by compile time extension of the basic strategy tag type. For the new tag type all instantiations of nodes that differ from the extended strategy have to be specialized. This leads to a new transformation of the expression tree.

3. Related Libraries

The most established C++ library for linear algebra is the matrix template library MTL2 [13]. Its architecture is similar to the STL. Matrices and vectors are defined as parameterized containers. For the optimization it uses the basic linear algebra instruction set [7] and a template library for small fixed size objects in particular. Template meta programs are used to reorder or unroll loops, the operations are executed step by step, expression templates are not used. There is only one strategy for evaluation, the standard way.

A complete reorganisation of the MTL is currently on the way. The MTL4 will implement a cursor concept instead of iterators, i.e. a strict separation of accessing the data, their position in the sequence, and the way of traversing it. In contrast to the array notation the cursor concept can store compile time data and use these for optimization. With respect to the evaluation of expressions there are no important differences to MTL2.

The boost ublas library [6] on the other hand uses expression templates but mainly to provide an interface for each subexpression that is comparable with that of an STL container. The actual operation is decoupled from the structure, a functor template parameter can be instantiated. This construction allows for defered evaluation of expressions. But a general choice of a backend is not possible, there is only one strategy.

Closest to our approach is the generic linear algebra system GLAS [9]. Expression templates are used to encode expressions and delegate their evaluation to different backends. The evaluation strategies do not belong to the data containers, but are specified via a function call applied to the left hand side of the assignment. In contrast to PTL strategies cannot be combined.

Acknowledgement

We wish to thank the referees for their detailed and helpful comments,

References

[1] D. Abrahams, B. Dawes. Boost. *http://www.boost.org*, 2005.

[2] G. Bohlender. Floating-point computation of functions with maximum accuracy. *IEEE Trans. Computers*, C26 No 7:621–632, 1977.

[3] J. Coplien. Curiously recurring template patterns. *C++ Report*, 7(2):24–27, 1995.

[4] D. Gregor and B. Stroustrup. Concepts *http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2081.pdf*

[5] R. Hammer et al. *C++ Toolbox for Verified Computing*. Springer, 1995.

[6] M. Koch and J. Walter. Boost ublas. *http://www.boost.org/libs/numeric/ublas/*, 2000.

[7] A. Lumsdaine and J.G. Siek. A rational approach to portable high performance: The basic linear algebra instruction set (blais) and the fixed algorithm size template (fast) library. *Parallel/High-Performance Object-Oriented Scientific Computing (POOSC)*, 1998.

[8] D. Marsden and J. de Guzman. Fusion 2.0. *http://spirit.sourceforge.net*, 2005.

[9] K. Meerbergen, T. Knappen. Generic linear algebra system. *http://glas.sourceforge.net/*, 2006.

[10] D. Knuth. The Art of Computer Programming *Addison-Wesley, 1969*

[11] T. Ogita, S. Oishi, S.M. Rump. Accurate floating-point summation. *Technical Report 05 1, Hamburg University of Technology*, 2005.

[12] S.M. Rump, S. Oishi, T. Ogita. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1995–1988, 2005.

[13] J. Siek. Matrix template library 2. *http://www.osl.iu.edu/research/mtl/*, 1998.

[14] T. L. Veldhuizen. Expression templates. *C++ Report*, 7(5):26–31, June 1995. Reprinted in C++ Gems, ed. Stanley Lippman, 1998.

[15] J. Wolff v. Gudenberg. Reliable expression evaluation in Pascal-SC. In R. E. Moore, editor, *Reliability in Computing*, pages 81–98. Academic Press, 1988.

Originally released as an electronic publication on CD.
IEEE Computer Society, 2006
Order Number E2821

ISBN-13: 978-0-7695-2821-2
ISBN-10: 0-7695-2821-X

Library of Congress Number 2007929345