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]

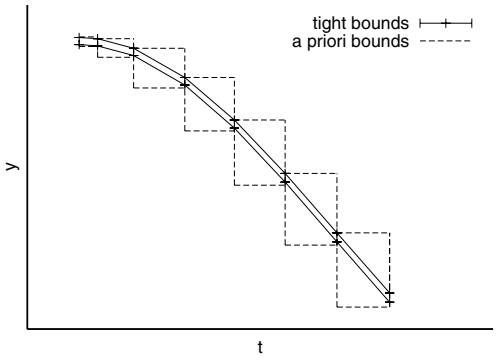

Figure 1. A priori and tight bounds.

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].

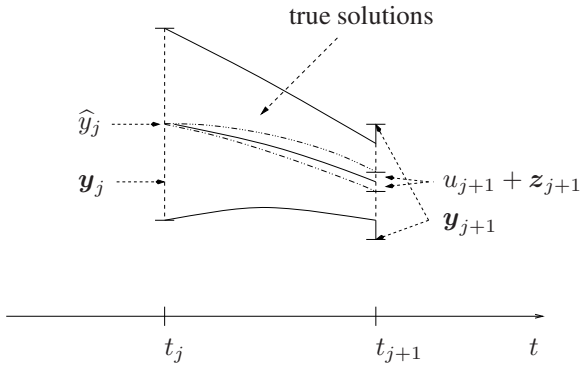

Figure 2. The enclosure at t_{j+1} is formed from an approximate point solution, an enclosure on the local excess, and an enclosure on the propagated global excess.

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

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

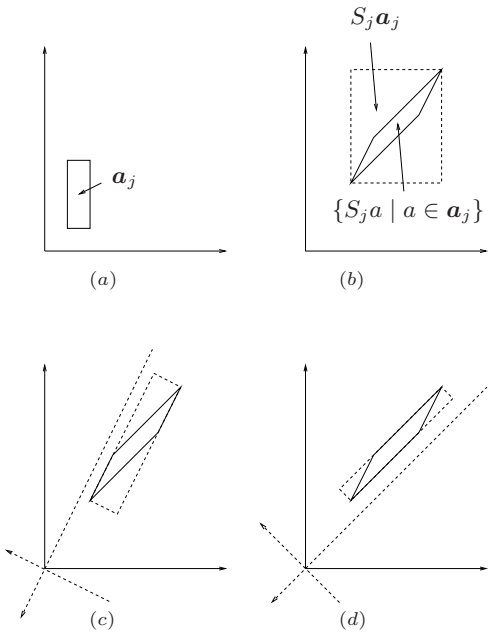

Figure 3. (a) $\boldsymbol{a}_j = \boldsymbol{y}_j - \widehat{y}_j$; (b) wrapping in the original coordinate system; (c) and (d) wrapping in a moving orthogonal coordinate systems that matches one of the edges of the enclosed set.

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.

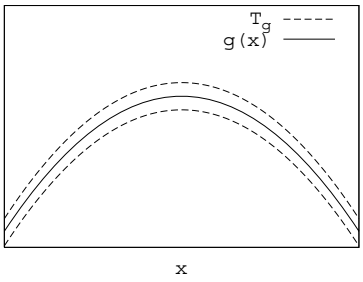

Figure 4. Function and its Taylor model enclosure.

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

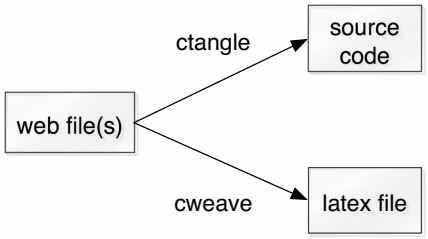

Figure 5. Producing C++ and LaTeX files from web files

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.

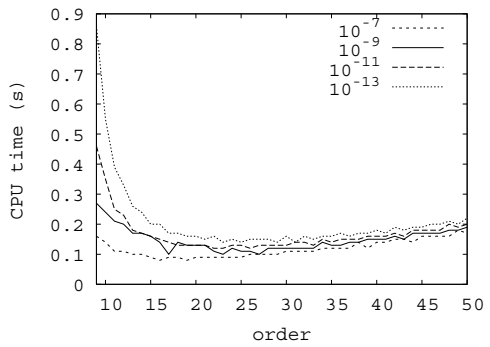

Figure 6. Work versus order for the Lorenz system.

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.

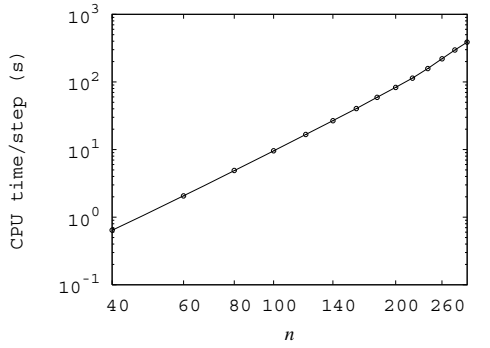

Figure 7. CPU time versus n for the DETEST C3 problem.

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

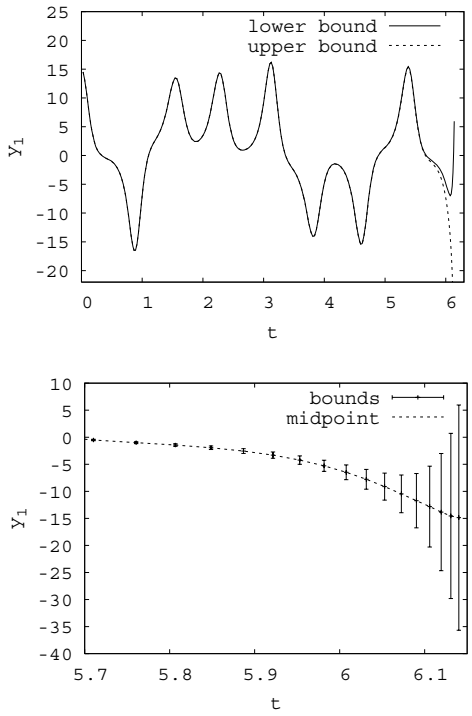

Figure 8. Bounds on y_1 versus t for the Lorenz system (13) with (14).

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`.