UNIVERSITAT DE GIRONA
Departament d'Electrònica, Informàtica i Automàtica

THESIS

Application of Modal Interval Analysis to the simulation of the behaviour of dynamic systems with uncertain parameters

Joaquim Armengol Llobet

Co-director:
Dr. Josep Vehí
Universitat de Girona

Co-director:
Dr. Louise Travé-Massuyès
LAAS-CNRS
France

Girona, November 1999

Contents

List of Figures

List of Tables

List of abbreviations and symbols

BIAS	Basic Interval Arithmetic Subroutines
BLAS	Basic Linear Algebra Software
Ca∼En	Causal Engine.
FD	Fault Detection.
FDI	Fault Detection and Isolation.
FRenSi	Fuzzy Region Simulation.
FuSim	Fuzzy Qualitative Simulation
INTBLAS	Interval Basic Linear Algebra Software
MBDS	Model-Based Diagnostic System.
MIA	Modal Interval Analysis.
MIMO	Multiple Input, Multiple Output.
MIS	Modal Interval Simulator.
MIS_0	MIS that generates exact envelopes in some particular cases.
NIS	Numerical Interval Simulation.
NSIM	Numerical Simulator using Interval Methods.
ODE	Ordinary Differential Equation.
PA	Predictive Algorithm
PDE	Partial Differential Equation.

PE	Predictive Engine
PROFIL	Programmer's Runtime Optimised Fast Interval Library
QA	Qualitative Analysis
QSIM	Qualitative Simulator.
SIGLA/X	Sigma Lambda Group. SIGLA/X membership: Calm R., Estela M.R., Gardeñes E., Jorba L., Mielgo H., Sainz M.Á. and Trepat A.
SISO	Single Input, Single Output.
SQSIM	Semiqualitative Simulator
TA	Transition Analysis
VE	Vector Envisionment

Acknowledgements

I am very grateful to many people. It is impossible to mention all of them and I would not like to forget any of them. To avoid this, I will not enumerate their names. I think that each of them already feel my sincere gratitude.

In spite of this, I would like to acknowledge my family, specially my wife and my parents, for their support and for providing me with the opportunity to explore a few of the many interesting scientific fields.

I also would like to express my gratitude and thanks to the directors of my thesis.

I would like to thank the members of the Departament d'Electrònica, Informàtica i Automàtica and the members of the Departament d'Informàtica i Matemàtica Aplicada of the Universitat de Girona who have helped me in any way.

I am also grateful to the colleagues I had (and that I still have) at the Escola Tècnica Superior d'Enginyers Industrials de Terrassa.

And last but not least, many thanks to the researchers world-wide with whom I have been enjoying many fruitful discussions and collaborations. Special thank goes to the ones who have helped me reviewing this manuscript.

Abstract

The mathematical quantitative models are simplifications of the reality and hence the behaviour obtained by simulation of these models differ from the real one. The use of complex quantitative models is not a solution because in most cases there is some uncertainty in the real system which can not be represented using these models. A way to represent this uncertainty is by using qualitative or semiqualitative models. A model of this kind represents a set of models indeed.

The simulation of the behaviour of quantitative models provides a trajectory across time for each output variable. This can not be the result of the simulation when a set of models is used. In this case, one way to represent the behaviour is by means of envelopes. The exact envelope is complete, i.e. includes all the possible behaviours of the model, and sound, i.e. every point inside the envelope belongs to the output of at least one instance of the model.

The generation of such an envelope is usually a hard task that can be approached, for instance, using algorithms of global optimisation or consistency checking. For this reason, in many cases approximations to that envelope are obtained. A complete but not sound approximation to the exact envelope is an overbounded envelope, whereas a sound but not complete envelope is an underbounded envelope. These properties are assessed for several simulators for uncertain systems.

One possible use of the envelopes is for fault detection by means of analytical redundancy. In this case, the system is considered as faulty when the measured output of the system is outside the envelope. If the measured output is inside the envelope, the system may also be faulty, but this is not detectable due to the dynamics of the system. This is what would happen if the exact envelope was used. If the envelope used for fault detection is overbounded there

can be missed alarms: the system is faulty but it is not detected because the measured output remains inside the envelope. If the envelope is underbounded then there can be false alarms: the system is said to be faulty, although it is not, because the measure goes out of the envelope. Therefore, these properties of the envelopes are very important when they are used to detect faults.

The approach proposed in this thesis is the use of error-bounded envelopes. This approach consists in the simultaneous computation of an underbounded and an overbounded envelope. These two envelopes determine three zones: the inner zone included in the underbounded envelope, the intermediate zone between the two envelopes and the outer zone outside the overbounded envelope. If the measure is in the outer zone, the system is guaranteed to be faulty. If the measure is in the inner zone nothing can be said because either the system is not faulty or if the system is faulty it can not be detected. Finally, if the measure is in the intermediate zone the situation is one of the previous but better approximations to the exact envelope are needed to decide which one of them.

An iterative algorithm to compute the two envelopes has been developed. It computes better approximations at every iteration. The condition to stop can be either the situation of the measured output of the system or the distance between the two envelopes. It is based on an interval model, that is a model in which the parameters are interval values instead of real numbers, and uses Modal Interval Analysis, an extension of interval arithmetic.

If it is assumed that the parameters of the system are uncertain but time invariant, the envelopes at each time step have to be computed starting from the initial state. This makes the algorithm unusable for long simulations. An approach to this problem is the use of sliding time windows. In this case the parameters of the model are allowed to vary in time at a speed that depends on the length of the window. Either the measured output of the system (real or interval) or the error-bounded envelopes can be used for the initial state at the beginning of the window.

The application of this simulator to fault detection gives different results depending on the options that are used for the simulation. One of these options is the length of the window. The adequate length depends on many factors. To help the user to choose it, several window lengths can be used simultaneously.

This technique has been applied to several examples and real cases. Some of the results are presented.

Resum

Els models matemàtics quantitatius són simplificacions de la realitat i per tant el comportament obtingut per simulació d'aquests models difereix dels reals. L'ús de models quantitatius complexes no és una solució perquè en la majoria dels casos hi ha alguna incertesa en el sistema real que no pot ser representada amb aquests models. Una forma de representar aquesta incertesa és mitjançant models qualitatius o semiqualitatius. Un model d'aquest tipus de fet representa un conjunt de models.

La simulació del comportament de models quantitatius genera una trajectòria en el temps per a cada variable de sortida. Aquest no pot ser el resultat de la simulació d'un conjunt de models. Una forma de representar el comportament en aquest cas és mitjançant envolupants. L'envolupant exacta és complete, és a dir, inclou tots els possibles comportaments del model, i correcta, és a dir, tots els punts dins de l'envolupant pertanyen a la sortida de, com a mínim, una instància del model. La generació d'una envolupant així normalment és una tasca molt dura que es pot abordar, per exemple, mitjançant algorismes d'optimització global o comprovació de consistència. Per aquesta raó, en molts casos s'obtenen aproximacions a l'envolupant exacta. Una aproximació completa però no correcta a l'envolupant exacta és una envolupant sobredimensionada, mentre que una envolupant correcta però no completa és subdimensionada. Aquestes propietats s'han estudiat per diferents simuladors per a sistemes incerts.

Un dels possibles usos de les envolupants és per a detecció de fallades mitjançant redundància analítica. En aquest cas, es considera que hi ha una fallada en el sistema quan la mesura de la sortida del sistema està fora de l'envolupant. Si la mesura de la sortida està dins de l'envolupant, també pot haver-hi una fallada en el sistema, però que no es pot detectar degut a la dinàmica del sistema. Això és el que passaria si es fes servir l'envolupant exacta. Si l'envolupant que es

fa servir per detecció de fallades està sobredimensionada hi pot haver alarmes perdudes: hi ha una fallada en el sistema però no és detectada perquè la mesura de la sortida roman dins de l'envolvent. Si l'envolupant és subdimensionada llavors hi pot haver falses alarmes: es diu que hi ha una fallada en el sistema, tot i que no hi és, perquè la mesura surt fora de l'envolupant. Per tant, les propietats de les envolupants són molt importants quan es fan servir per a detectar fallades.

L'enfoc que es proposa en aquesta tesi és l'ús d'envolupants amb l'error acotat. Aquest enfoc consisteix en el càlcul simultani d'una envolupant subdimensionada i una envolupant sobredimensionada. Aquestes dues envolupants determinen tres zones: la zona interior inclosa en l'envolupant subdimensionada, la zona intermèdia entre les dues envolupants i la zona exterior fora de l'envolupant sobredimensionada. Si la mesura està en la zona exterior, està garantit que hi ha una fallada en el sistema. Si la mesura està en la zona interior no es pot dir res perquè o bé no hi ha cap fallada o si hi és no es pot detectar. Finalment, si la mesura està en la zona intermèdia la situació és una de les dues anteriors però cal millors aproximacions a l'envolupant exacta per a discernir entre elles.

S'ha desenvolupat un algorisme que calcula les dues envolupants. Aquest algorisme calcula aproximacions millors a cada iteració. La condició per parar les iteracions pot ser o bé la situació de la mesura de la sortida del sistema o bé la distància entre les dues envolupants. L'algorisme es basa en un model intervalar, és a dir un model en el qual els paràmetres tenen valors intervalars en comptes de números reals, i utilitza l'Anàlisi Intervalar Modal, una extensió de l'aritmètica intervalar.

Si se suposa que els paràmetres del sistema són incerts però invariants en el temps, les envolupants a cada instant de temps s'han de calcular a partir de l'estat inicial. Això provoca que l'algorisme no es pugui fer servir per simulacions llargues. Una possible solució a aquest problema és l'ús de finestres temporals lliscants. En aquest cas es permet que els paràmetres del model variïn a una velocitat que depèn de la longitud de la finestra. Es poden utilitzar tant les mesures de la sortida del sistema (reals o intervalars) com les envolupants amb error acotat com a estat inicial al principi de la finestra.

L'aplicació d'aquest simulador a la detecció de fallades proporciona resultats diferents segons les opcions triades per a fer la simulació. Una d'aquestes opcions és la longitud de la finestra.

La longitud de finestra adequada depèn de molts factors. Per ajudar l'usuari a triar-la, es poden utilitzar diverses longituds de finestra simultàniament.

Aquesta tècnica s'ha aplicat a diversos exemples i casos reals. Es presenten alguns dels resultats.

Chapter 1

Introduction

1.1 Motivation

The original motivation for the work described in this thesis arose, on one side, from the collaboration, in the years 1993-1994, of its author in the Esprit III project "TIGER : Real-Time Situation Assessment of Dynamic, Hard to Measure Systems" and, on the other side, from the development of the MIA (Modal Interval Analysis) by the SIGLA/X Group [30].

The goal of the TIGER Project [62, 82, 83] was the design of an automated mechanism to monitor the state of a gas turbine continuously. It focused on to gas turbines. One of the components of this mechanism was a MBDS (Model-Based Diagnostic System) which applied the principle of the analytic redundancy. In order to apply this principle, it is necessary to have a measurement and a prediction of a variable. The comparison between these two values indicates whether the system is behaving as expected or not.

The predicted value is obtained through a model of the system and prediction or simulation ("prediction" is used for a one step ahead prediction whereas "simulation" implies multi-steps prediction) of this model. The model of a system usually has uncertainties or imprecisions due to different reasons. In the TIGER Project these uncertainties were included in the model by expressing the parameters of the model with intervals instead of real numbers. The problem was then to simulate such a model.

This was done by the Ca∼En (Causal Engine) [81], which, among other modules, includes a simulator that generates envelopes. The envelopes provide an upper and a lower bound for the value of a variable at each sampled instant. Therefore, the comparison between the measured and the predicted value is done by indicating if the measurement is inside or outside of the predicted envelope. Although the detection mechanism runs on a one step prediction mode when the measurement is inside of the envelope, it switches to a simulation mode when it is not.

The problems were that in the simulation mode the envelopes were not tighten enough and moreover they were unstable. These are consequences of the simulation method implemented and of the use of an interval-based reasoning. To solve the former problem, the simulation method must be revised and enhanced, whereas the latter problem seems that can be solved using enhanced extensions of interval arithmetic in order to deal with intervals. One of these extensions is MIA, firstly introduced by Gardeñes [31] and now being under development at the Universities of Barcelona and Girona.

1.2 Goals of this thesis

The aims of this thesis are:

1. to critically examine existing simulators that can generate envelopes to identify their strengths and weaknesses,
2. to use the information thus gleaned to design and construct a new simulator based on MIA and
3. to test the resulting simulator in the domain of FD (Fault Detection) with examples and real applications.

1.3 Summary of original contributions of this work

The original work contained in this thesis consists of the following items: the analysis of existing envelope generators, the design and production of the different versions of the MIS (Modal Interval Simulator) and the experimental testing of the features of MIS.

The novel features of each of these are as follows:

- A critical and comparative analysis of some existing envelope generators with regard to the properties of the envelopes they generate.

- The design of a new simulation technique arising from the analysis of the existing simulators. Its novelty comprises the following aspects:

  - A branch and bound algorithm based on MIA that computes interval extensions of functions with adequate semantic properties. This algorithm is used to determine approximations to the range of functions in a parameter space.

  - The use of an implementation of modal intervals arithmetic which uses directed roundings.

  - The use of this algorithm for the generation of error-bounded envelopes.

  - The generation of envelopes using sliding time windows.

  - The use of multiple sliding time windows.

  - The implementation in Matlab to facilitate its future integration into a supervision framework that is being developed based on Matlab and Simulink [87].

- Application of these simulators to FD in academic and real examples.

1.4 Thesis organisation

The work contained in this thesis is in the following parts. The first part consists of chapters 2 and 3. It involves a presentation of the problem of the simulation of the behaviour of uncertain systems, making special emphasis in the use of envelopes to represent this behaviour, and an overview of the simulators that can be used for this task. The second part consists of chapters 4 and 5, which introduces the new simulators that have been developed. A third part consisting of chapters 6 and 7 in which the simulators are applied and conveniently adapted for FD. Finally, chapter 8 summarises this work and gives conclusions and directions for the future work.

The following are the summaries of the individual chapters:

Chapter 2 presents the problem of the simulation of the behaviour of uncertain systems and some of the approaches that have been used. The approach used in this work is based on the use of interval models, which include a representation of the uncertainty, and envelopes. The properties of the envelopes are also discussed.

Chapter 3 provides an overview of the different techniques that have been applied to the simulation of uncertain systems. A description of some simulators is presented and the properties of the envelopes that each of them generate are outlined and compared. These simulators use many different techniques that range from the pure qualitative ones to the pure quantitative ones and between these two ends there are many semiqualitative methods.

Chapter 4 presents a simulator for interval models based on the range computation method with a new approach: the use of MIA, which is also introduced. The advantages and disadvantages of this method, which intends to obtain the exact envelopes, are discussed.

Chapter 5 shows that in many cases it is not necessary to obtain the exact envelopes. Hence, a new simulator that computes error-bounded envelopes is introduced and its properties are discussed.

Chapter 6 shows one of the applications of envelopes: FD. One of the approaches to FD consists in comparing the output of a system with a reference one generated through simulation and the faults are detected using the discrepancies. This technique is situated in its context, formed by the model-based analytical redundancy methods. Error-bounded envelopes are applied to FD.

Chapter 7 presents applications of the new simulators to real cases. This shows their strengths and weaknesses.

Chapter 8, finally, contains and extended summary that outlines this work. It also provides some general conclusions and introduces some lines for the future work.

Chapter 2

Simulation of uncertain systems

2.1 Introduction

This chapter presents the problem to model and simulate uncertain systems. Quantitative models do not represent the uncertainty of the systems, so other kinds of models (qualitative, semiqualitative) are needed. The results of the simulation of these qualitative and semiqualitative models (envisionments and envelopes among them) and the results of the simulation of quantitative models are also different. Envisionments and envelopes are presented.

2.2 Models

A model is an abstraction of a system. It describes some of the characters of the system that are of interest for the work to be done. Therefore, a model is a simplification of the real world and is not unique, i.e. different models for different goals can be obtained from a single system. There are many kinds of models [20]:

- Physical models. Scale models, analog computers. An analog computer is based on systems analogy and allows to create an electric system that behaves the same as a given system which can be of a very different nature (mechanical, chemical, thermal, etc.).
- Mental models. An understanding of how a particular system works.
- Symbolic models:

- Not mathematical:
  - Linguistic models. A linguistic description of the behaviour of a system under different circumstances.
  - Graphical models. Diagrams, graphs, charts.
  - Schematics models. Trees, networks, task-flow charts.
- Mathematical.

This work is based on mathematical models, which describe the relations between the variables of the system (the state) by means of mathematical equations. There are many types of mathematical models, which can be classified attending to the characteristics of these mathematical equations:

- Static or dynamic.
- Linear or not linear.
- Lumped parameter or distributed parameter.
- Continuous-time or discrete-time (including discrete-event).
- Deterministic or probabilistic (stochastic).

For example, a dynamic model may describe the behaviour of a system by means of differential (continuous-time) or difference (discrete-time) equations. ODE (Ordinary Differential Equations) can be used for a lumped parameter model while PDE (Partial Differential Equations) are used for distributed parameter models.

The mathematical models can also be classified attending to the way they have been obtained:

- Modelling. The model is obtained studying the physical laws that affect to the system.
- Identification. Inputs and outputs are known but the system is assumed to be a black box, i.e. its contents are unknown. The model is a mathematical function that relates the output to the input.

2.3 Simulation

To simulate the behaviour of the system means to predict the sequence of future states using the model [14] or, in other words, to reproduce the evolution of the system across time by means of the model [15]. In the case that ODEs are used for the model, traditionally this task has been performed integrating analytically the differential equations in order to obtain time functions for the variables of the system. The problem is that an analytical solution can be obtained only in relatively simple cases [16]. Another option is the use of digital computers to integrate numerically these equations.

There are many numerical integration algorithms with different degrees of complexity and, hence, different degrees of precision in the results. Some of them are Euler, Runge-Kutta, Adams, Gear, linsim, LSODE, etc. [45, 4]. All of them make time a discrete variable and approximate temporal trajectories of variables with straight lines. Usefulness of each algorithm depends on the characteristics of the system: linearity, stiffness, sampling time, etc.

Nowadays, there are many simulators and simulation languages based on these algorithms. They can be classified according to different characteristics:

- Field of application. In all-purpose or general simulators, models are expressed using differential equations, state space models or transfer functions, so they can be used for any kind of system: chemical, mechanical, electrical, etc. Examples are Matlab-Simulink [4] and ACSL [1]. On the other hand, there are specialised simulators. For instance, SPICE [8] is a simulator for electrical systems. It includes libraries with models of electric and electronic components. These components can be combined into circuits that can be analysed by means of simulation. There are similar simulators for other fields like chemistry, hydraulics, mechanics, etc.

- Model characteristics [14]. Many simulators deal only with continuous-time or discrete-time models. There are also simulators for discrete-event models like Witness [9] and Microsaint [5]. Finally, there are hybrid simulators, which can combine all these types of models, like ACSL or Shift [7].

- Model reusability. Many simulators can not reuse the models due to causality problems. For instance, a model of a resistor is a gain R if the input is the current and the output is the voltage and a gain $\frac{1}{R}$ in the opposite case. Nevertheless, object-oriented simulators allow reusing the models. Some simulators of this kind are Dymola [2] and Omola [6]. These models are grouped into libraries and simple models can be combined to create more complex models. Important active research is being done in this field. A great effort must be done to finish their definition and, above all, to create the libraries of models.

2.4 Models of uncertain systems

In most of the cases there is a difference between the real behaviour of a system and the predicted one. These differences are due to the sensors (the measures are only approximations to the real values) and to the model of the system, because usually it is not very accurate although it is precise [64]. This inaccuracy of the model is intrinsic, as it has been seen above, because a model is an approximation of the reality. In some cases, it is possible to have an accurate model of the system but it is too complex and a simplified one is more appropriate for the task to be undertaken [15]. This happens, for instance, when a non-linear system is linearised around an operating point and, therefore, a linear model is used to represent it. This happens, too, when a low order model is used to represent the behaviour of a higher order system in a range of frequencies.

However, in many cases there are uncertainties or imprecisions that make difficult, if not impossible, to obtain accurate models. Some sources of this inaccuracy of the models are:

- The knowledge of the system is not complete because the real system can not be observed or does not exist yet (a factory before it is built or a product before it is manufactured) [15, 89].
- There are physical phenomena that are difficult to identify or to predict [86].
- The parameters of the system can change across time due to unknown, unpredictable or difficult to model phenomena [89].

These uncertainties can be unstructured (the equations that model the system are not known) or structured (the equations are known but the values of their parameters are not known). They can not not be represented with quantitative models, i.e. models in which the parameters are real numbers. If it is necessary to consider these uncertainties, other kind of models are needed to overcome this shortcoming of the quantitative models. Some types of models that can represent the uncertainty of the systems are qualitative, fuzzy and interval models.

2.4.1 Interval models

Consider the example of a crane. Its behaviour is different for different loads and different lengths of the rope. The model structure remains the same but its parameters change if the load or the rope length change. If there were a finite set of possible different loads and rope lengths, this system could be represented by a discrete set of quantitative models. As this set is not finite, a model space is needed [48]. This model space represents uncertainties or imprecisions explicitly and can be expressed by an interval model, i.e. a model in which there are parameters represented by intervals instead of real numbers. For instance, such an interval model could be given by the following differential equation:

$$A_n \frac{d^n y}{dt^n} + A_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \ldots + A_1 \frac{dy}{dt} + A_0 y = x \tag{2.1}$$

where:

- A_i are the interval parameters of the model:

$$A_i = [\underline{a_i}, \overline{a_i}] \tag{2.2}$$

- x is the input to the system.
- y is the output of the system.
- t is the time.

Therefore, an interval model is in fact a family, a class or a space of models [14] and a quantitative model can be viewed as a particular interval model in which interval widths are zero. Hence, as interval widths decrease, the precision of the model increases [49]. An interval model is also referred to as a semiqualitative model [89] because it allows an easy integration of qualitative and quantitative knowledge. For instance, the ranges of the parameters can be defined by the experts.

An interval model is useful when the uncertainties are structured, i.e. the structure of the model is known and only the parameters of the equations undergo imprecisions or uncertainties. This happens in many real cases, in which a real value for a particular parameter can not be determined but it can be bounded.

This kind of model can be used in multiple cases:

- To represent in a single model the behaviour of a system in different situations.
- To represent time variant systems. The parameters can vary across time following a known or unknown time function. In both cases, each parameter can be bounded by an interval including all possible values.
- To model complex processes with simple interval models [89].

2.5 Representations of the behaviour of uncertain systems

When the behaviour of a quantitative model is assessed using an adequate simulator, the result is a single trajectory across time for each variable. This can not be the result of the simulation of the behaviour of a semiqualitative or a qualitative model, which represents some uncertainty. The behaviour of such a model can be represented in different ways. In the following, the envisionments and the envelopes are presented. They are complementary representations. Depending on the particular application one of them will be more appropriate than the other and, in some cases, both of them should be used.

2.5.1 Envisionment

The behaviour of the system is represented globally using a graph like the one in figure 2-1 [23, 49]. This graph is a tree in which all the possible behaviours of a system starting from an initial state are qualitatively represented. It is a tree and not a single path because of the propagation of the uncertainty [14, 15, 16]. There are different types of envisionments depending on whether the initial state is specified or not:

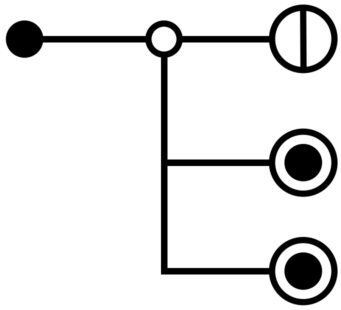


Figure 2-1: Envisionment.

- Total envisionment [28]. When the initial state is not specified. For instance, if there is a tank with an input flow and an output flow, the level can increase, decrease or stabilise depending on the relation between these two flows. Hence, three different behaviours can follow one single initial state (the tank with two non-zero flows).

- Complete envisionment [56]. When the initial state is specified, that is, initial values of exogenous variables (inputs and outputs) are known but initial values of endogenous variables (internal variables, state variables) are unknown. In the previous example of the tank, if the input flow is zero and the output flow is not zero, only one of the three behaviours is possible.

One of the envisionment simulators is VE (Vector Envisionment) [69]. QSIM, that will be presented in section 3.4.1, and some simulators derived from QSIM can produce envisionments, too.

Envisionments will not be used in this work and only envelopes will be considered.

2.5.2 Envelope

In geometry, an envelope is defined as the curve or surface that is tangent to each member of a system or curves or surfaces, the form and position of the members of this system being allowed to vary according to some continuous law.

In the context of simulation, the system of curves is formed by all the possible behaviours of an uncertain system starting from a specific initial state, precise or not. These curves are trajectories across time and, according to the definition of envelope used in geometry, there are two envelopes in this case: the upper one and the lower one. In simulation, both of them are considered as a whole and are referred to as an envelope.

Hence, an envelope represents all the possible behaviours of an uncertain system in one single image. It splits the set of possible values of a specific output variable at a time point into two subsets: the allowed values and the forbidden ones according to the model and the input applied to the system.

Envelopes can be obtained using different techniques: methods derived from numerical integration algorithms, qualitative reasoning, consistency check, fuzzy logic, etc. Some authors call attainable envisionments [28] or partial envisionments [56] to the envelopes obtained by means of qualitative simulation.

2.6 Properties of the envelopes

An envelope, according to its definition, includes all the possible behaviours of the model given an input and every point inside the envelope belongs to the output of at least one instance of the model, i.e. one of the quantitative models that belong to the space of models. Therefore, it is complete and sound.

The generation of the envelope is not a realistic goal in most cases. It is difficult or even impossible, so in many cases approximations to it are obtained. Then, in a broader sense, a complete (resp. sound) approximation to the envelope is referred to as a complete (resp. sound) envelope. From now onwards the complete and sound envelope is referred to as the exact envelope.

It is worth noticing that this is the terminology used in [80]. Some authors use the same words in the opposite sense [92].

Definition 1 *An envelope is overbounded if it is complete but not sound.*

An overbounded envelope is wider than the exact one: all the possible outputs of the models that belong to the model space are inside the envelope but there are points inside the envelope that do not belong to any of these outputs and hence includes zones that can not be reached by any of the models belonging to the family.

Definition 2 *An envelope is underbounded if it is sound but not complete.*

An underbounded envelope is tighter than the exact one, so all points inside the envelope belong to at least one of the outputs of the models belonging to the model class but there are points of these outputs outside the envelope. Therefore, there are models belonging to the set whose output is outside the envelope, hence the envelope does not represent their behaviour.

These properties are summarised in figure 2-2.

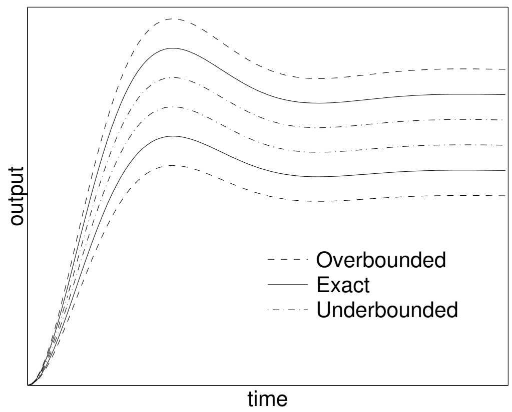


Figure 2-2: Properties of the envelopes.

2.7 Summary

In this chapter, the problems related to modelling and simulation of uncertain systems have been presented. There are many techniques to perform the simulation of different types of quantitative models. This is a problem already solved. The problem is that usually there are uncertainties and then the model is only an approximation of the system. These uncertainties can be represented in the model by means of qualitative or semiqualitative models. The behaviour of these models can be represented, for instance, by envisionments or envelopes.

The properties of the envelopes have been discussed in this chapter.

Next chapter provides an overview of the different techniques that have been applied to simulate the behaviour of uncertain systems.

Chapter 3

Simulators for uncertain models: an overview

3.1 Introduction

The use of a quantitative model and its simulation by means of an appropriate simulator is enough to solve many problems. Nevertheless, there are other cases where the knowledge is incomplete and a whole set of models must be considered. The simulation algorithms for quantitative models can not be used in these cases [16] and, therefore, appropriate simulation techniques must be used. A lot of research on the simulation of models which include a representation of the uncertainty has been done in the fields of systems engineering and artificial intelligence [16]. It is one of the hardest tasks of systems supervision because each one of the quantitative models included in a qualitative or semiqualitative one has a different behaviour [17, 49]. The way to deal with uncertainty depends on the available information (numeric, qualitative, etc.) and the future use of the results. There are many methods: qualitative, statistics, Monte Carlo, interval, error treatment, etc. All of them try to make robust simulations. This means to predict the behaviour of all the quantitative models that belong to the model space [48].

This chapter presents a survey of different simulators that can be used to deal with qualitative and semiqualitative models focusing on the generation of envelopes [11]. These simulators use many different techniques that range from the pure qualitative ones to the pure quantitative ones and between these two ends there are many semiqualitative methods. Semiqualitative simulators combine characteristics of the previous two types and its classification can be subdivided into different groups, as shown in figure 3-1. It can be seen in this figure that there are semiqualitative simulators built starting from a qualitative simulator to which quantitative modules that perform interval or fuzzy set calculus have been added. Other semiqualitative simulators are not based on a qualitative simulator.

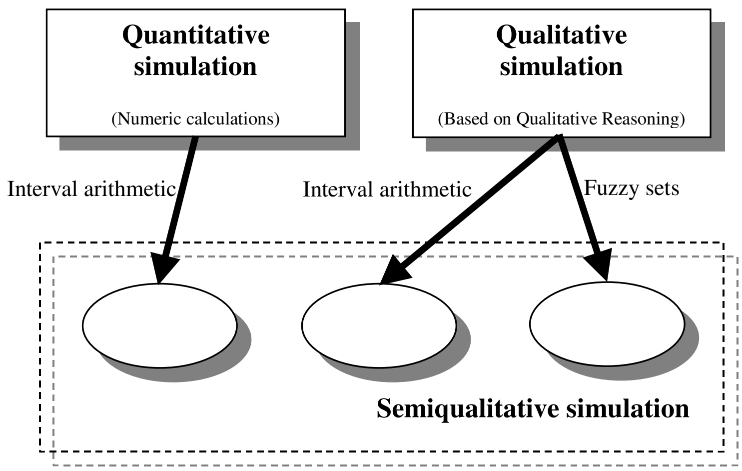


Figure 3-1: Classification of simulators.

A description of some of these simulators is presented and the properties of their respective envelopes are outlined and compared.

3.2 Quantitative simulation

Quantitative or numeric simulation makes numeric predictions of the system states. This implies the prediction of the values of the variables at determined time points.

The following provides a description of different quantitative methods to simulate the behaviour of interval systems.

3.2.1 Threshold calculus

This technique is based on the passive generation of residuals described in section 6.4.1. The envelope is generated by superimposing a threshold (fixed or adaptive) to the trajectory of the nominal system, obtained using a simulator for quantitative models.

The model can be obtained, for instance, using parameter identification. In this case, mean, variance, probability distribution, etc. are obtained for each parameter. Then the nominal system can be a system in which every parameter takes its mean value and the threshold can be computed, for instance, through variance propagation [49]. This threshold represents an estimation of the error at each point or the likelihood or confidence degree of the envelope at each time point.

Therefore, the results actually can not be considered as envelopes so neither completeness nor soundness can be guaranteed.

One disadvantage of these methods is that they are applicable only if the system is linear, accessible, measurable and the uncertainty can be modelled as a probability [49, 15]. Another disadvantage is that it is necessary to deal with great amounts of data to model uncertainty as a probability distribution.

3.2.2 Quantitative models method

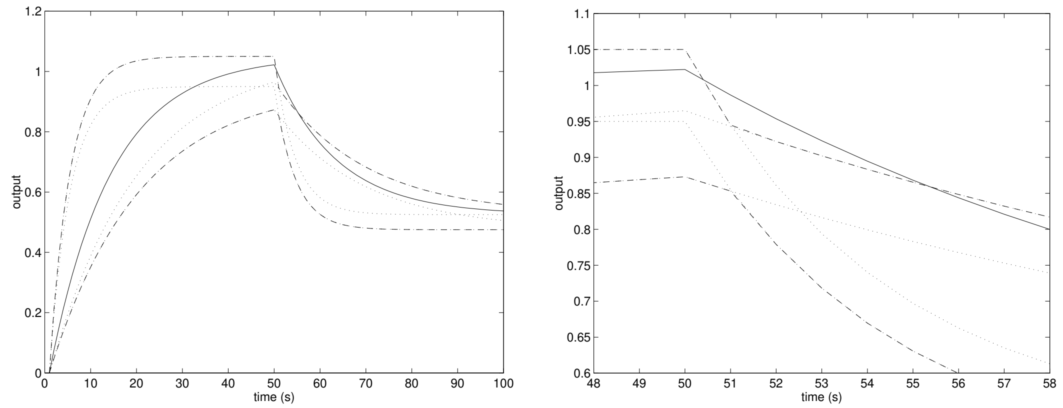
Another way to study a family of models is studying many quantitative models belonging to the family and, then, extracting conclusions for the whole family. Using these techniques, sometimes it is not possible to extrapolate the properties of the studied models to the whole family. Not studied models to which the extrapolation is not valid may exist [49]. This is the reason to say that these methods do not have guarantee. This will be demonstrated with a practical case in the next paragraphs.

In the case of simulation, models belonging to the family must to be chosen and simulated. The envelopes are obtained by superimposing all the trajectories generated by the simulations, which can be performed with any of the existing simulators for quantitative models.

An intuitive choice of the quantitative models to be simulated can be the extreme models, the ones obtained combining the ends of each interval. For instance, given the interval model $F(s) = \frac{k}{\tau s+1} = \frac{[0.95,1.05]}{[5,20]s+1}$ the extreme models are $F_1(s) = \frac{0.95}{5s+1}$, $F_2(s) = \frac{0.95}{20s+1}$, $F_3(s) = \frac{1.05}{5s+1}$

and $F_4(s) = \frac{1.05}{20s+1}$.

The output of these models when the input is 1 between $t = 0$ s and $t = 50$ s and 0.5 after this time point, is shown in figure 3-2a. In this figure, the outputs of the four extreme models are represented with dotted lines and the resulting envelope in dashed lines. Moreover, the output of the model $F_5(s) = \frac{1.05}{14s+1}$ is represented in solid line. This model belongs to the considered family but its output is, for some period of time, outside the extreme models envelope, as it can be seen in the zoom of figure 3-2b.

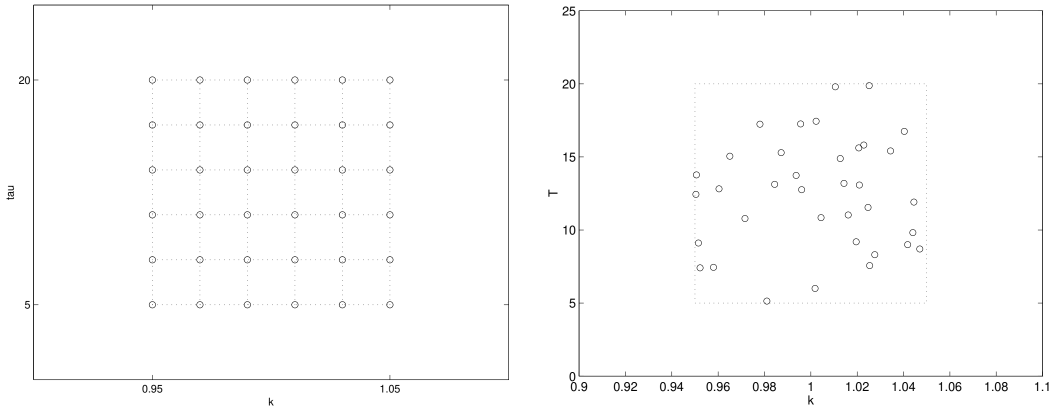


(a) (b)

Figure 3-2: (a) Simulation of the extreme models and another quantitative model and (b) zoom.

The envelope obtained simulating only the extreme models is sound but not complete hence underbounded. It can be widened simulating other models belonging to the family. These additional models can be chosen systematically by making a grid in the parameter space, as it is shown in figure 3-3a, or randomly (Monte Carlo), as it can be seen in figure 3-3b. In the latter case, the possibility to chose a particular quantitative model can be conditioned or not by the probability distribution of each parameter.

However, the result is always of the same kind: the widened envelope will be closer to the exact one but it remains underbounded. An unstudied quantitative model whose output is outside the envelope can still exist. Hence, this is a method with no guarantee of completeness: conclusions can be taken out from the studied quantitative models, but not from the unstudied ones. Nevertheless, a good property of this method is convergence: as the number of studied



(a) (b)

Figure 3-3: Quantitative models chosen (a) systematically and (b) randomly.

quantitative models increases, the envelope converges towards the exact one. In the limit, if infinity quantitative models belonging to the family could be simulated, the obtained envelope would be the exact one [14].

A limitation of this method is that it is valid only for time invariant systems because time variant systems can not be simulated. Finally, this method is not recursive and it can not be used in real time. The reason is that it is necessary to simulate the behaviour of all the considered quantitative models from time point 0 to time point t and search for the maximum and the minimum outputs in order to compute the bounds of the envelope at a specific time point t.

3.2.3 Range computation

The recurrence problem can be solved using a discrete representation of the system. For instance, assume that the behaviour of a n-th order dynamic SISO (Single Input, Single Output) system is represented by the following difference equation:

$$y_t = \sum_{i=1}^{m+1} a_i y_{t-iT} + \sum_{j=1}^{p+1} b_j u_{t-jT} \tag{3.1}$$

in which it can be observed that the output of the system at any time point (y_t) depends on the values of the previous outputs (y_{t-iT}) and inputs (u_{t-jT}), being T the sampling time. This dependency is given by the parameters of the system model (a_i and b_j). The output can have different values if any of the other variables or parameters is an interval. The maximum and the minimum of these values determine the envelope. This will happen in the following cases:

- The model is an interval one. In this case, there are parameters of the model which value is an interval.
- The input of the system is an interval. This is the case if there are uncertainties in the actuator or in a sensor that is used to measure the input.
- The initial output y_0 is an interval.

In this case the difference equation can also be seen as the expression of a function into a parameter space. For instance, the difference equations at the steps n and $n+1$ of a discrete-time model are:

$$\begin{aligned} y_n &= \sum_{i=1}^{m+1} a_i y_{n-i} + \sum_{j=1}^{p+1} b_j u_{n-j} \\ y_{n+1} &= \sum_{i=1}^{m+1} a_i y_{n-i+1} + \sum_{j=1}^{p+1} b_j u_{n-j+1} \end{aligned} \tag{3.2}$$

The parameter space has the shape of a hypercube and its number of dimensions is the sum of interval parameters appearing in the function: outputs, inputs and parameters of the model. Therefore, the computation of the exact interval output, that is the output envelope, is equivalent to finding the maximum and the minimum of the function in that parameter space. This is a range computation problem.

One way to solve this problem is by using a global optimisation algorithm. Some simulators that are presented in the next sections are based on this conversion of a simulation problem into a range computation one.

The parameters of the model and some previous inputs and outputs appear in both difference equations of (3.2). If the range of y_n and the range of y_{n+1} are calculated independently, it may happen that the value of a parameter that maximises y_n is not the same that the one that maximises y_{n+1}. This means that these parameters may take different values at each step of the simulation and hence it is considered that the system is a time variant one.

There are cases where it is known that the parameters have a fixed value, although it is not precisely known due to some uncertainty. In this case the system is time invariant and hence the value of each parameter must remain the same at each step of the simulation. The method presented above does not take this into account. It considers the function at each time step as independent when there is a dependency between them.

One way to consider that every time that a parameter appears in different equations is not an independent parameter is by merging these equations into a single expression. Then, the expression for the step n is obtained in a recursive way by substituting y_n within the previous equation down to y_0. For instance, in the simplest case the following expressions are used:

$$\begin{aligned}
y_1 &= a_1 y_0 + b_1 u_0 \\
y_2 &= a_1^2 y_0 + a_1 b_1 u_0 + b_1 u_1 \\
y_3 &= a_1^3 y_0 + a_1^2 b_1 u_0 + a_1 b_1 u_1 + b_1 u_2 \\
&\vdots \\
y_n &= a_1^n y_0 + b_1 \sum_{i=1}^{n} a_1^{i-1} u_{n-i}
\end{aligned} \tag{3.3}$$

The range of y_i in these expressions has to be determined in order to obtain the exact envelopes for time invariant systems. These envelopes are tighter than the ones for time variant systems due to the dependency that now is taken into account. The overview presented in this chapter considers that the systems are time invariant. Hence the properties of the envelopes generated by the simulators are given comparing their envelopes with the exact ones based on this consideration.

In the case of this method, the properties of the envelopes depend on the particular application as it is a generic method. For instance, if these ranges are computed by means of a global optimisation method that provides local optima, the envelopes are underbounded.

This method of simulation based on the range computation is the one used in the work presented in this thesis. A similar option is the re-formulation of the simulation problem as a set of optimal control problems with fixed final time [26]. An analytic solution is obtained for linear systems. Although it is difficult to use in the practical case because the solution is not finite, it provides a reference to the exact solution.

3.3 Semiqualitative simulation based on quantitative simulation and interval arithmetic

Interval arithmetic allows to consider the whole range of possible instances represented by an interval model. It hence guarantees that the envelopes are not underbounded [35].

Methods that use interval arithmetic are defined as semiqualitative methods because they use more than numeric knowledge.

3.3.1 Introduction to interval arithmetic

Computers work with digital numbers, not with real ones. To express a real number with a digital one, it is truncated and rounded. Therefore, it is only an approximation of the real number. When an arithmetic operation is performed, there is some imprecision in the result and when many operations are performed the results can be not only imprecise but also incorrect. It may seem that a possible solution to this problem is an increase of the number of digits. It may seem that another possible solution is to compute twice with different number of digits. Then, if the results coincide until a particular digit, these digits are correct. Hansen [36] shows that this is not correct by using an example:

Example 1 *The value of the function*

$$f\left(x,y\right)=333.75y^{6}+x^{2}\left(11x^{2}y^{2}-y^{6}-121y^{4}-2\right)+5.5y^{8}+\frac{x}{2y} \tag{3.4}$$

is computed for $x = 77617$ *and* $y = 33096$ *with a S/370 computer. Input data do not have representation problems. The only errors are due to truncation and rounding. The obtained values obtained using single, double and extended precision, which correspond approximately to 6, 17 and 34 digits, are:*

- *Single precision:* $f = 1.172603\ldots$

- *Double precision:* $f = 1.1726039400531\ldots$

- *Extended precision:* $f = 1.172603940053178\ldots$

The first seven digits after the decimal point are equal. However, all of them are incorrect. Even the sign is incorrect. The correct result is $f = -0.8273960599468213$. This has been tested using Maple [3]. The correct result is obtained using 37 or more digits but even in this case there is not guarantee on the result. Some examples of the consequences of these erroneous computations are the explosion of the Ariane 5 and the failure of a Patriot missile during the Gulf War [12].

Interval arithmetic is defined over a set of intervals instead of being defined over a set of real numbers. Whereas computations performed using a floating-point representation are a mere estimate of the correct result, interval-based computing methods provide an interval and guarantee bounds for this correct result, considering truncations and rounding conveniently. The advantage of interval arithmetic is not just a matter of checking computing errors; it also offers ways to reason on the range of values of variables.

There are older antecedents, but it can be considered that the main ideas about interval computations appear for the first time in [66]. In his Ph. D. thesis, R.E. Moore studied the errors caused by truncation and rounding in arithmetic operations performed using digital computers. The first monograph on interval analysis [67] is the starting point of interval analysis.

Nowadays, interval analysis is mostly developed in USA and Germany. Good introductions to interval analysis can be found in [10] and [68].

Intervals

An interval is defined as a set of real numbers $\{x|\underline{x} \leq x \leq \overline{x}\}$ and is noted $X = [\underline{x}, \overline{x}]$, in which $\underline{x}$ is the infimum and $\overline{x}$ is the supremum. Interval numbers are an extension of the real numbers, as a real number x is the degenerate interval $[x, x]$. The interval arithmetic rules, which will be shown in 3.3.1, are simpler when they are used with degenerate intervals.

The set of closed intervals of $\mathbb{R}$ is $I(\mathbb{R}) = \left\{[a,b]' \mid a, b \in \mathbb{R}, \quad a \leq b\right\}$.

Interval arithmetic

The addition of the intervals $X=[\underline{x},\overline{x}]$ and $Y=\left[\underline{y},\overline{y}\right]$ is the set $\{x+y|x\in X,y\in Y\}$. This is also an interval: $Z=[\underline{z},\overline{z}]=\left[\underline{x}+\underline{y},\overline{x}+\overline{y}\right]$. In a similar way, the different arithmetic operations can be defined. If **op** denotes an arithmetic operation for real numbers, the corresponding interval arithmetic operation is

$$Z = X\ \mathbf{op}\ Y = \{x\ \mathbf{op}\ y \mid x\in X,\ y\in Y\} \tag{3.5}$$

The interval Z, result of the arithmetic operation X **op** Y, consists of all the possible values that can be obtained operating every $x\in X$ with every $y\in Y$.

This definition is applied to obtain the rules of the arithmetic operations starting from the ends (infimum and supremum) of the intervals:

- Addition:

$$X+Y=\left[\underline{x}+\underline{y},\ \overline{x}+\overline{y}\right] \tag{3.6}$$

- Difference:

$$X-Y=\left[\underline{x}-\overline{y},\ \overline{x}-\underline{y}\right] \tag{3.7}$$

- Product:

$$X*Y=\left[\min\left(\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\right),\ \max\left(\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\right)\right] \tag{3.8}$$

which can be broken down depending on the signs of the ends of each interval factor:

$$X*Y=\begin{cases}
\left[\underline{x}\underline{y},\ \overline{x}\overline{y}\right] & \text{if } \underline{x}\geq 0 \text{ and } \underline{y}\geq 0\\
\left[\overline{x}\underline{y},\ \overline{x}\overline{y}\right] & \text{if } \underline{x}\geq 0 \text{ and } \underline{y}<0<\overline{y}\\
\left[\overline{x}\underline{y},\ \underline{x}\overline{y}\right] & \text{if } \underline{x}\geq 0 \text{ and } \overline{y}\leq 0\\
\left[\underline{x}\overline{y},\ \overline{x}\underline{y}\right] & \text{if } \underline{x}<0<\overline{x} \text{ and } \underline{y}\geq 0\\
\left[\overline{x}\overline{y},\ \underline{x}\overline{y}\right] & \text{if } \underline{x}<0<\overline{x} \text{ and } \overline{y}\leq 0\\
\left[\underline{x}\overline{y},\ \overline{x}\underline{y}\right] & \text{if } \overline{x}\leq 0 \text{ and } \underline{y}\geq 0\\
\left[\underline{x}\overline{y},\ \underline{x}\underline{y}\right] & \text{if } \overline{x}\leq 0 \text{ and } \underline{y}<0<\overline{y}\\
\left[\overline{x}\overline{y},\ \underline{x}\underline{y}\right] & \text{if } \overline{x}\leq 0 \text{ and } \overline{y}\leq 0\\
\left[\min\left(\overline{x}\underline{y},\underline{x}\overline{y}\right),\ \max\left(\underline{x}\underline{y},\overline{x}\overline{y}\right)\right] & \text{if } \underline{x}<0<\overline{x} \text{ and } \underline{y}<0<\overline{y}
\end{cases} \tag{3.9}$$

- Division:

$$\frac{1}{Y} = \left[\frac{1}{\overline{y}}\,,\,\frac{1}{\underline{y}}\right] \text{ if } 0 \notin Y \tag{3.10}$$

$$\frac{X}{Y} = X * \left(\frac{1}{Y}\right) \text{ if } 0 \notin Y \tag{3.11}$$

- Interval to the power of a real number:

$$X^n = \begin{cases} [1,1] & \text{if } \; n = 0 \\ [\underline{x}^n,\; \overline{x}^n] & \text{if } \; \underline{x} \geq 0 \text{ and } n \text{ is even, or if } n \text{ is odd} \\ [\overline{x}^n,\; \underline{x}^n] & \text{if } \; \overline{x} \leq 0 \text{ and } n \text{ is even} \\ [0, \max(\underline{x}^n,\; \overline{x}^n)] & \text{if } \; \underline{x} \leq 0 \leq \overline{x} \text{ and } n \text{ is even} \end{cases} \tag{3.12}$$

Addition and product have the associative and commutative properties. Unfortunately, the inverses for the sum and the product do not exist, in general, and they do not always fulfil the distributive property. It is fulfilled only in some cases. For instance:

- If x is real and Y and Z are intervals: $x\,(Y+Z) = xY + xZ$.
- If $Y \cdot Z > 0$: $X\,(Y+Z) = XY + XZ$.

In the rest of the cases only a subdistributivity is fulfilled:

$$X\,(Y+Z) \subseteq XY + XZ. \tag{3.13}$$

Therefore, interval arithmetic is a generalisation of real numbers arithmetic. If interval operations are performed between degenerate intervals, the properties are the ones of the arithmetic of real numbers.

Interval functions

Given a real function f of real variables $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ which belong to the intervals $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^T$, the ideal interval extension of f would be a function that provides the exact range of f in the domain $X_1, X_2, \ldots, X_n$.

If $\mathcal{D} \subseteq I\!R^n$, then $\mathbf{I}(\mathcal{D}) \triangleq \{\boldsymbol{X} \in \mathbf{I}^n \;|\boldsymbol{X} \subseteq \mathcal{D}\}$ is the set of all the hypercubes included in $\mathcal{D}$.

Definition 3 *The united extension* $R_f(\boldsymbol{X}) : \mathbf{I}(\mathcal{D}) \to \mathbf{I}(\mathbb{R})$ *of a continuous function* $f : \mathcal{D} \to \mathbb{R}$ *is*

$$R_f(\boldsymbol{X}) \triangleq \left[\min_{x \in \boldsymbol{X}} \{f(x_1, ...x_n)\}, \max_{x \in \boldsymbol{X}} \{f(x_1, ...x_n)\} \right] \tag{3.14}$$

in which R_f *is the range of the function* f *into* $\boldsymbol{X} \in \mathbf{I}(\mathcal{D})$.

In the general case this extension is hardly computable. Hence the goal of interval arithmetic is to find computable interval extensions which include the exact range of the function R_f. In the following, the definition of interval extension and the needed properties to allow the computation of overbounded approximations of R_f are given.

An interval function is an interval value that depends on one or several interval variables. Consider f as a real function of the real variables $x_1, x_2, \ldots, x_n$ and F as an interval function of the interval variables $X_1, X_2, \ldots, X_n$.

Definition 4 *The interval function* F *is an interval extension of* f *if* $F(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) \quad \forall x \in \mathcal{D}$.

Therefore, if the arguments are degenerate intervals the result of computing $F(x_1, x_2, \ldots, x_n)$ must be the degenerate interval $[f(x_1, x_2, \ldots, x_n), f(x_1, x_2, \ldots, x_n)]$. This definition assumes that interval arithmetic is exact. In practice, there are rounding errors and the result of computing F is an interval that contains $f(x_1, x_2, \ldots, x_n)$:

$$f(x_1, x_2, \ldots, x_n) \in F(X_1, X_2, \ldots, X_n) \tag{3.15}$$

To compute the range of the function f, it is not enough to have an interval extension F. Moreover, F must be an inclusion function and must be inclusive monotonic.

Definition 5 *Let* $R_f(\boldsymbol{X})$ *be the range of the function* f *into* $\boldsymbol{X} \in \mathbf{I}(\mathcal{D})$. *A function* $F : \mathbf{I}(\mathcal{D}) \to \mathbf{I}$ *is an inclusion function of* f *if* $R_f(\boldsymbol{X}) \subseteq F(\boldsymbol{X}) \quad \forall \boldsymbol{X} \in \mathbf{I}(\mathcal{D})$ *and* $F(x) = f(x) \quad \forall x \in \mathcal{D}$

Hence the range obtained computing the interval extension $F(\boldsymbol{X})$ is an overbounded approximation of the exact range of f into $\boldsymbol{X}$.

Definition 6 *An interval function is inclusive monotonic if* $X_i \subset Y_i \ (i = 1, 2, \ldots n)$ *implies* $F(X_1, X_2, \ldots, X_n) \subset F(Y_1, Y_2, \ldots, Y_n)$

Therefore, if $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 = \boldsymbol{X}$, an inclusive monotonic function verifies

$$R_f(\boldsymbol{X}) \subseteq F(\boldsymbol{X}_1) \cup F(\boldsymbol{X}_2) \subseteq F(\boldsymbol{X}) \tag{3.16}$$

The usual procedure to compute the range of a function f into a specified domain $\boldsymbol{X}$, consists of finding an inclusive monotonic function and computing an overbounded approximation of the range. Better approximations can be obtained splitting the domain.

Therefore, inclusive monotonic functions for real function have to be found.

When f is a rational function, the natural extension $FR(\boldsymbol{X})$ can be defined.

The natural extension of a rational function f is obtained by substituting each real variable by the corresponding interval one and the rational operations by the corresponding interval ones [68].

An interval arithmetic operation is inclusive monotonic according to the definition given by equation (3.5). Therefore, if $FR(\boldsymbol{X})$ is a rational interval function, i.e. F includes only additions, differences, products and divisions, then F is inclusive monotonic. Moreover, it has been proved that the natural extension $FR(\boldsymbol{X})$ is an inclusion function of f over $\boldsymbol{X}$:

$$R_f(\boldsymbol{X}) \subseteq FR(\boldsymbol{X}) \tag{3.17}$$

Hence, overbounded approximations of $R_f(\boldsymbol{X})$can be obtained simply computing the natural extension by means of interval arithmetic.

Example 2 *The exact range of the function*

$$y = x^4 - 10x^3 + 35x^2 - 50x + 24 \tag{3.18}$$

in the parameter space $X = [0, 4]$ *is* $Y = [-1, 24]$. *The overbounded approximation to the range that is obtained using the natural extension is*

$$Y = X^4 - 10X^3 + 35X^2 - 50X + 24 = \tag{3.19}$$

$$= [0,4]^4 - [10,10] \cdot [0,4]^3 + [35,35] \cdot [0,4]^2 - [50,50] \cdot [0,4] + [24,24] = [-816, 840]$$

Most of the functions used in computer science and in applied mathematics can be expressed combining basic arithmetic functions. Interval rational inclusion functions of irrational functions can also be found, including basic functions like the trigonometric ones, exponential, logarithms, etc. An important research line in interval arithmetic [68] intends to find rational inclusion functions of irrational functions that allow to compute overbounded ranges as tight as possible.

Therefore, the general procedure to compute the range of an interval function consists of obtaining a rational inclusion function which is computed using interval arithmetic.

Computation of rational interval functions

Interval arithmetic can be used to compute rational interval functions, that is interval functions in which only arithmetic operations are used.

The main limitation of interval arithmetic is that it has not some of the properties of real number arithmetic, for instance the distributive property. This means that the exact range of a function is not always computable. However, the natural interval extension has some interesting properties:

- Reliability. $FR(\boldsymbol{X}) \supseteq R_f(\boldsymbol{X})$.
- The exact range $R_f(\boldsymbol{X})$ is obtained if there are not multi-incident variables in $FR(\boldsymbol{X})$.

Definition 7 *A variable is multi-incident in a function if it appears more than once in the expression of the function.*

Therefore, the determination of the exact range of a function is a problem only when there are multi-incident variables, because each incidence is considered as an independent variable. In this case, this problem is similar to a global optimisation one [71, 72].

Example 3 *The range of the function*

$$f(x) = \frac{x}{x+1} \tag{3.20}$$

in the parameter space $X = [2, 3]$ *is* $\left[\frac{2}{3}, \frac{3}{4}\right]$ *according to the graphical representation of the function in figure 3-4, where it can be observed that it is monotonic in the interval* X. *If the* *range of the function is calculated using its natural extension and interval arithmetic*

$$F(X) = \frac{[2,3]}{[2,3]+1} = \frac{[2,3]}{[3,4]} = \left[\frac{2}{4}, \frac{3}{3}\right] = \left[\frac{1}{2}, 1\right] \tag{3.21}$$

which is overbounded as

$$\left[\frac{2}{3}, \frac{3}{4}\right] \subseteq \left[\frac{1}{2}, 1\right] \tag{3.22}$$

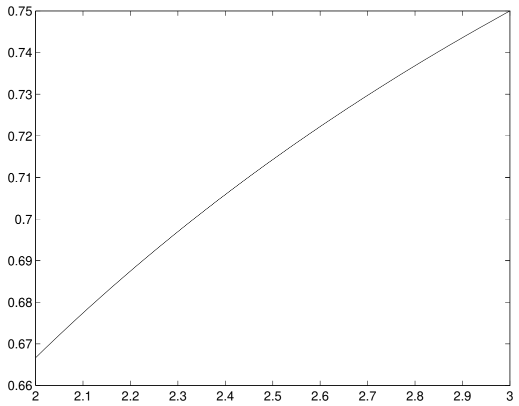


Figure 3-4: $f(x) = \frac{x}{x+1}$ for $2 \leq x \leq 3$.

The research on interval arithmetic has produced many techniques to compute the range of functions. Their main interest is the guarantee on the results, which always are overbounded approximations of the exact ones. The precision can be fixed to the desired value. Among these techniques there are numeric and algebraic ones, Bernstein polynomials, etc.

Numeric algorithms Inclusive monotonic extensions of interval functions produce overbounded approximations to their range. According to (3.16), better approximations can be obtained splitting the domain of the function. The parameter space of the function, a hypercube, is split into subspaces smaller and smaller and the result is closer and closer to the exact one. In some cases, to split a subspace into subsubspaces does not produce a better result and hence it is not necessary to do it. Therefore it is possible to do branch cutting in these cases, which is based on the study of the properties of the function: first and second derivatives, etc. Many algorithms apply both techniques simultaneously. Details and examples about these branch and bound algorithms can be found in [36].

Algebraic techniques As a consequence of the subdistributivity property of interval arithmetic (3.13), the results of computing the range of functions by means of interval extensions are highly dependent on the syntactic expression of the function.

Example 4 *For instance, the function* $f(x) = x^2 - x$ *can also be written* $f(x) = \left(x - \frac{1}{2}\right)^2 - \frac{1}{4}$*. The natural extension of the first one in the parameter space* $X = [0,2]$ *gives an overbounded approximation of its range:*

$$F_1(X) = X^2 - X = [0,2]^2 - [0,2] = [-2,4] \tag{3.23}$$

while the natural extension of the second function provides the exact range:

$$F_2(X) = \left(X - \tfrac{1}{2}\right)^2 - \tfrac{1}{4} = \left([0,2] - \tfrac{1}{2}\right)^2 - \tfrac{1}{4} = \left[-\tfrac{1}{4}, 2\right] \tag{3.24}$$

The reason is that in $F_1(X)$*, the variable* X *is multi-incident and in* $F_2(X)$ *it is uni-incident.*

Therefore, a way to obtain tighter, but always guaranteed, results is expressing the function with the lowest number of multi-incidences. Many techniques search for the best way to express a function to obtain the tightest results [71]. Some of them are [86]:

- cascade evaluation [37, 38].
- alternative expressions for polynomials [78, 79] like:
  - Horner forms.
  - centred forms [67]:
    - mean value form.
    - Baumann form.
    - Taylor forms.
  - Bernstein polynomials:
    - for univariate polynomials [19, 73].
    - for multivariate polynomials [32, 33].
- adequate inclusion functions for rational functions like $\sin(x)$, $\exp(x)$ or $\sqrt{x}$ [68].

Implementations of interval arithmetic

The limits a and b of a given interval $[a, b]$ may not be represented by a computer. In this case it must be rounded outwards, i.e. a must be rounded to the biggest representable number smaller or equal to a, and b must be rounded to the smallest representable number greater or equal to b. This requires direct rounding, that is rounding upwards or downwards depending on what is needed. This is direct rounding, which is specified in IEEE 754 standard on floating point arithmetic as an option. It was first implemented in the microprocessor Intel 8087 and is essential for interval arithmetic implementations in order to guarantee the results [46].

One of the first implementations of interval arithmetic is INTLIB [51]. It was developed first in Fortran 77 and after in Fortran 90. It consists of a set of subroutines and functions for basic interval operations based on BLAS (Basic Linear Algebra Software). Direct rounding is simulated. This makes it portable but increments computation time. As a consequence, the precision of the computations can be fixed to the desired value.

PROFIL/BIAS [54] is probably the most complete and fastest interval library. It is written in C++ and supports many common interval and real operations. Data types are INT, REAL, INTERVAL and works with vectors and matrices of these types and also with complex numbers. PROFIL (Programmer's Runtime Optimised Fast Interval Library) is based on BIAS (Basic Interval Arithmetic Subroutines), which was developed starting on BLAS. BIAS intends to provide an interface for interval operations. Its goals are to take advantage of the hardware, portability and independence of the specific interval representation. PROFIL/BIAS is a public domain software in continuous evolution. Project INTBLAS (Interval Basic Linear Algebra Software) intends to enhance BIAS for instance improving the memory access. Researchers from 10 countries participate in this project. The software is being implemented in C++ and Fortran 90.

RVInterval [85] is a public domain interval arithmetic package implemented in C++. It is much more simpler than PROFIL/BIAS and also slower, but can be useful for small applications.

IN C++ is a commercial software by Delisoft implemented in C++ as well [39].

Commercial environments for linear algebra like Mathematica or Maple V also include commands to perform interval computations, but automatic simplification of symbolic expressions and the lack of direct roundings make them obtain bad results in some cases. INTPAK [24, 25] is a package for Maple V that overcomes some of these defects.

Finally, Interval Solver [42, 40, 41, 43] is a commercial package that allows Microsoft Excel to work with intervals. It is the first interval arithmetic implementation for a standard commercial spreadsheet. Interval Solver uses cascade evaluation [37, 38] to compute the range of functions.

Applications of interval arithmetic

There are applications of interval arithmetic in many fields, mainly since there are implementations of interval arithmetic. One of these fields is global optimisation [36]. The computation of the range of a function into a parameter space is a problem of global optimisation with constraints. However, according to the point of view of interval arithmetic it is a problem of global optimisation without constraints.

Other fields are astrophysics, geology, image processing, robotics, navigation, spectral analysis, computer graphics [84], etc. Fields in which data are imprecise or obtained by indirect measurement. In [52] there are described applications of interval arithmetic to global optimisation algorithms, solving of linear systems, fuzzy logic, economic models, quality control, medical expert systems, etc. A field where there are many applications is systems engineering: semiqualitative simulation (the subject of this thesis), system identification [75], analysis and design of robust controllers for uncertain systems [86, 88], etc.

Conclusions

This section has introduced interval arithmetic and shows that it has reached a certain degree of maturity. It has been implemented in different platforms and programming languages and some of the software has even been commercialised. However, interval arithmetic applications have the limitations inherent to interval arithmetic. Hence, often the results are much overbounded and if tighter results are needed, high computational efforts are required. Moreover, as the results are overbounded, conclusions can not be drawn in all cases.

3.3.2 Use of interval arithmetic for simulation

There are some semiqualitative simulators based on interval arithmetic and monotonic inclusion. Some of them assume that the model is quantitative or nearly precise and the imprecision is due to arithmetic errors in the computer or to the rounding errors that are produced when the real numbers are translated to floating point numbers [49]. These simulators use numeric integration algorithms revised for intervals, so they can easily become general and the simulations are fast. However, the results are highly overbounded if there is a bigger imprecision in the models [16, 48, 49].

The first interval simulator was the one of Moore [67, 68]. This simulator works with k-order Taylor expansions of the differential equations of the system model and with the Euler integration algorithm [48]. The envelopes obtained are much overbounded because it ignores multi-incidences. In other words, when there are multi-incident variables, part of the available information is not used.

Moreover, the wrapping problem appears. This problem is linked to the use of interval arithmetic and was described for the first time by Moore [67]. Consider a system described by a set of variables. Since every variable has an interval value, the system's state at some time point may be represented by an hypercube. However, it may be that the system's state does not evolve into another hypercube at the next time point. In figure 3-5 an example is shown in which there are two state variables and, therefore, the hypercube in the parameter space is a rectangle. This rectangle evolves to a rhombus (it could evolve to any figure in two dimensions) in the following time step. As the value of each variable is still expressed with an interval, the new state is represented with a new rectangle that includes all possible states (the rhombus) but also spurious states, shown in shadow in the figure. Hence, the obtained envelopes will be overbounded and, possibly, unstable. This problem can not be solved by the use of smaller steps in the simulation, because more steps will be needed.

Other simulators based on these algorithms are the interval simulator by Markov and Angelov [61] and the simulator AWA [58]. The latter tries to minimise the wrapping effect by moving the system's co-ordinates along the trajectory of the system in the parameter space. This goal is achieved in some cases [49].

The description of two simulators based on interval arithmetic, NSIM and NIS, follow.

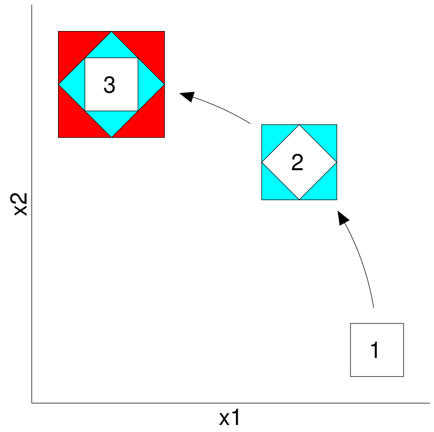


Figure 3-5: The wrapping problem.

NSIM

NSIM (Numerical Simulator using Interval Methods) [47, 48, 49] is based on QSIM (see section 3.4.1). It uses the same models than QSIM and has inherited a limitation from it: NSIM works only with monotonic functions.

NSIM is an enhancement of Moore's interval simulator when there exists quasi-monotonicity. It is said that a system is quasi-monotonic if the exact envelopes can be obtained by simulating the extreme quantitative models (see section 3.2.2) [49].

This simulator generates static envelopes. Given an interval differential equation (a differential equation in which the parameters are intervals), it searches for a pair of functions (extreme functions) which bound the original function in a certain domain. Hence, completeness is guaranteed. After this, a simulator for quantitative models is used for the simulation. This is one of its advantages: it is not necessary to develop a specialised simulator and, therefore, it makes good use of speed and efficiency of existing simulators and exploits their possible enhancements.

The disadvantage is that if there are multi-incident or correlated variables, the extreme functions do not belong to the initial family of functions and, therefore, they are outside its domain. For instance, the function $f(x) = \frac{x}{x+1}$ has the following extreme functions:

- Upper function: $\overline{f(x)} = \frac{\overline{x}}{\underline{x}+1}$
- Lower function: $\underline{f(x)} = \frac{\underline{x}}{\overline{x}+1}$

which do not belong to the family of functions $f(x)$.

The results of the simulation are overbounded envelopes due to ignored correlations and to the wrapping problem. These envelopes are tighter than the ones obtained with the simulators described in [68, 61], but they can diverge and become unstable with certain oscillatory systems.

To solve these problems, Kay proposes various ways:

- Wrapping problem. To define the intervals with respect to a mobile system of co-ordinates instead of using a fixed one. In other words, he proposes to work into the error state space. This implies the computation of Jacobean and sensitivity matrices. Research must be done to determine if this can be done when the available differential equations are imprecise. It is not clear if an imprecise Jacobean matrix gives sufficient information to work into the error state space.

- Ignored dependencies. To build models without ignored dependencies. That is, decrease the number of multi-incident variables and their number of incidences.

- Split state space into smaller subspaces. As it was noted in section 3.3.1, when there are multi-incident variables, the result of evaluating the range of the functions using interval arithmetic is proved to be closer to the exact range.

As a conclusion, the problem of interrelations between variables is still unsolved by NSIM.

NSIM has inherited from QSIM an advantage, too. This advantage is the qualitative representation, which is used to generate different envelopes for qualitatively different behaviours.

NIS

NIS (Numerical Interval Simulation) [89] is an extension of the fuzzy simulation methods for linear systems proposed in [90, 91]. It works with state variables. Therefore, the ODE used are first order ones. They can be non-monotonic, must be continuous in the considered parameter space and can contain the following operations: sum, difference, product, division, exponentiation, logarithm, sine, cosine and constants. The parameters of the model and the initial values of the state variables can be intervals.

Some simulators use interval arithmetic to compute, at every time step of the simulation, the maximum and the minimum values of the derivative of each variable. Then these values are used for Euler or Runge-Kutta integration algorithms. These simulators obtain highly overbounded envelopes due to ignored multi-incidences. In order to avoid this source of overbounding, NIS computes the maximum (resp. minimum) value of the derivative at the current time step only in the point where the function has its maximum (resp. minimum) value in this step. The consequence is that it is not complete, because, as it is shown in section 3.2.2, not necessarily the maximum value at a time point is a consequence of the maximum value at the previous time point. There can be a point where the maximum derivative is greater than the one of the maximum point and hence it determines the maximum point in the following time step.

Therefore NIS is based on the same principle than NSIM: it generates a pair of functions (upper limit of the envelope and lower limit of the envelope) given a function with incomplete knowledge and then it simulate these functions numerically. The difference is that NSIM generates the extreme functions before the simulation and NIS uses interval arithmetic at every simulation step.

NIS also can become unstable with certain oscillatory systems.

Gasca's simulator

As stated in section 3.3.2, one way to avoid the wrapping problem when using interval arithmetic is to define the intervals with respect to a mobile system of co-ordinates, instead of defining them with respect to a fixed one. This can be done by computing the trajectory of one of the quantitative models that belong to the family under study and then superimposing a threshold to it (see section 3.2.1). This simulator [35, 34] is based on this principle. The models are represented by band functions and the parameters are represented by intervals covering all the real values corresponding to each qualitative label. The trajectories of the family of systems are represented by polyhedrons in the state space defined from a point, a transformation matrix and an uncertainty vector.

Given that the transformation matrix is obtained after some simplifications and approximations, the properties (completeness, soundness, stability) of the resulting envelopes are not known. They are generally unsound because when using interval arithmetic multi-incidences are not taken into account and incomplete because sometimes it predicts a single value at a particular time point.

3.4 Qualitative simulation

Artificial Intelligence proposes different non-probabilistic formalisms to deal with uncertainty: theory of possibility (based on fuzzy sets theory), evidence theory and qualitative reasoning [16]. To perform qualitative reasoning it is necessary to have a qualitative model. Some motivations for developing qualitative models are [64]:

- to provide simpler computational mechanisms.
- to provide a description for systems where traditional methods are ineffective.
- to provide modelling paradigms that accord more closely with common sense intuition of the operation of physical systems.
- to develop modelling methods based on the principles of knowledge based systems.

The main approaches to qualitative modelling are constraint, component, and process centred approaches [64], which lead to concepts like quantity space, landmark, sign, magnitude, etc. The model expresses qualitative information about the relations between the variables. For instance, if the value of the variable a increases, the value of the variable b also increases.

The information about the variables, from a purely qualitative point of view, can be a positive value, a negative value or zero (there is a fourth possibility to be considered: the unknown value). Hence, the real axis is divided into three parts. This information is used by an inference engine, generally simple [48], and some conclusions are obtained [23]. These tools for reasoning are very useful to solve problems in which the information is poor, as is the qualitative knowledge, but given that the information is poor, the results are poor as well. If there is more information, it can be used, for instance, introducing more landmarks and hence dividing the quantity space in more than three parts. But there are still notable deficiencies in the attempts that have been made to take advantage of a richer information. If the complexity increases, the predictions become poorer [16] and the computational effort to be made increases as well.

The qualitative simulation makes a prediction of the qualitative states in which the system will be using a non-numeric model. It distinguishes qualitatively (with labels, not with numerical values) the states where the system will be or the values that the variables will have. The qualitative simulators are used when the knowledge about the system has important limitations or when it is interesting to have qualitative results.

Most of the qualitative simulators are not causal and hence do not consider time. Therefore, they predict which states will exist in the future but neither when they will occur nor how long they will last. Some qualitative simulators, however, are causal and consider time.

The qualitative simulation consists in two phases [23]. In the first one, called TA (Transition Analysis), many transitions are generated without using the model of the system. In the second phase, called QA (Qualitative Analysis), these transitions are filtered using the model. Hence, the transitions that do not fulfil the constraints of the model are eliminated. This is called constraint propagation.

A criterion to classify qualitative simulators is the degree of constructivity [23]. A quantitative simulator is constructive because it uses the model of the system to generate the values of the variables of the system: it solves the equations and, if it is a dynamic system, it integrates them. In other words, it "constructs" the future values of the variables starting from the model of the system. On the other hand, a qualitative simulator is not constructive because the model of the system is used only to filter out the impossible states that have been generated before. Semiqualitative simulators are somewhere between these two extreme points, according to the degree of use of the system model. If a simulator is more constructive than another one, it is more efficient in order to take advantage of the available information and hence it generates less impossible states [93]. In spite of this, in some cases a non-constructive simulator is very useful. For instance, when there are algebraic loops a non-constructive simulator can be used but a constructive simulator can not.

Some qualitative simulators (QSIM, PA and Ca$\sim$En) are described below. They are only a sample of the existing ones: SQUALE [63], DIAPASON [65], etc.

3.4.1 QSIM

QSIM (Qualitative Simulator) is the most popular qualitative simulator. Since it was first implemented [56], many new functions have been added continuously. Hence, probably it is also the most sophisticated simulator, due to its constant evolution [23].

Each variable is represented by its value and the value of its derivative. The value of the derivative is expressed in a purely qualitative way: *inc* for increasing (positive derivative), *dec* for decreasing (negative derivative) or *std* for steady (zero derivative, stationary variable). The value of the variable can be expressed also in a purely qualitative form or, if more information is available, its value space can be split in more than three parts (labels) using landmarks.

The model of a system is represented by the relations between its variables. These relations can be algebraic operations (sum, product, sign change), derivative (one variable is the derivative of another one), monotonicity (if the value of a variable increases, the value of the other increases too), etc.

The algorithm of simulation consists of three parts: transition rules, constraint and Waltz filters, and global filter.

The transition rules are tabulated. There are two types of transition rules: P and I. As each variable is represented with a pair magnitude/derivative, the temporal function of the value of the variable is approximated with a first order Taylor series. Therefore, the transition rules are based on the Euler numeric integration algorithm [23]. They are based on the intermediate value theorem and the mean value theorem. The intermediate value theorem defines the direction of the change of the value of a variable between two time points. The mean value theorem indicates that in a continuous system it is compulsory to pass through zero to change the sign [23].

The transition rules are applied to each variable individually, as if each variable was independent from the others. In consequence, impossible transitions are generated. In the subsequent phase, some of these impossible transitions are eliminated because they do not fulfil the constraints. QSIM is not a constructive simulator because it does not use the model of the system to generate the transitions.

The result of the simulation is a representation of the qualitative states (expressed by the variables' values and the values of their derivatives) that will succeed to the present state. The duration of these states is not given because QSIM is not causal and hence it does not consider time.

The original QSIM was complete [56], as it found all the possible behaviours of the system. However, new filters, which optionally can be used or not, have been added through the years and some of them make QSIM to lose this property [23]. On the other hand, it is not sound because it can find spurious behaviours. One of the reasons is that the constraints must contain only two or three variables. If there are actual constraints with more than three variables, they must be decomposed into other constraints introducing intermediate variables in order to introduce them into QSIM. A value space is assigned to these intermediate variables. Therefore, if a predicted state coincides totally or partially with various labels, QSIM considers that the variable can take any of the values included in these labels, including the ones that have not been predicted.

Other characteristics of QSIM are that the only non-monotonic function allowed is the product and that inputs like sinus, for instance, are not allowed as they could produce a combinatorial explosion [89].

In the following sections, it will be shown that this simulator has been the starting point of many other simulators: Q2, Q3, NSIM, SQSIM, PA, FuSim, Mycroft, etc.

3.4.2 PA

PA (Predictive Algorithm) [93] is a qualitative simulation algorithm integrated in PE (Predictive Engine) [94]. It belongs to the same family as QSIM but it is more constructive than QSIM. The problem of QSIM is that in the TA phase of the simulation, many impossible transitions are generated and some of them can not be eliminated in the QA phase. One solution is to restrict the generation of impossible transitions in the TA phase, making it more constructive, more similar to a quantitative simulator.

In PA, the user chooses the number of successive derivatives to be used for each variable. QSIM only uses the value of each variable and the value of its first derivative. If the user wants to use higher order derivatives, new intermediate variables must be created and these variables will be considered independently in the first phase of the simulation. For example, if somebody wants to use the second derivative of the variable x, the first derivative must be defined as a new variable $y = \frac{dx}{dt}$ which will be represented with its value and the value of its derivative $\frac{dy}{dt} = \frac{d^2x}{dt^2}$.

Therefore, PA has more flexibility to represent the model of the system, which can be interpreted as a set of differential planes. Hence, there are two planes in QSIM and the desired number of planes can exist in PA [23]. Such a description of the model implies a set of constraints that are used to generate the states of the variables. The procedure to generate the transitions begins at the plane of the highest order derivatives and decreases the order of the derivatives until it ends at the plane with no derivatives (zero order derivatives). Therefore, PA already uses the system model in the TA phase of the simulation, while QSIM uses it only in the QA phase for filtering. That is the reason why PA is more constructive than QSIM.

Another aspect that makes PA more constructive than QSIM is that in QSIM the order of the equations is not important and, in the TA phase of the simulation, all variables are considered independent. However, equations must be causally ordered in PA so less impossible transitions are generated in the TA phase.

In spite of that, PA is not totally constructive and hence the temporal information obtained from the simulation is insufficient.

In the phase of filtering, the generated transitions are checked for consistency using the constraints. If a transition does not fulfil all the constraints, it is eliminated. If a transition fulfils all the constraints, the transitions still unverified are eliminated and the simulation continues hence ignoring other possible transitions. This makes PA incomplete.

3.5 Semiqualitative simulation based on qualitative simulation and interval arithmetic

As a conclusion, neither quantitative techniques nor qualitative ones (they do not take advantage of the available numerical information) are appropriate for interval systems simulation. For this reason, it can be said that interval systems simulation is still an unoccupied gap [15].

Quantitative information must be added to qualitative methods in order to enhance them. This will reduce the uncertainty of the qualitative state variables [16]. This can be done, for instance, by discretising the real axis hence splitting it in more than three parts. Therefore, a new value space, which is more complex than the pure qualitative one, is defined. The discretisation can be predefined, fixing the existing qualitative values a priori, or not, allowing any subinterval. In the former case, the qualitative knowledge is represented easily and it can be difficult to represent the quantitative knowledge properly. The latter case is the opposite to the former one because the representation of the quantitative information is favoured [17]. The simulators described in section 3.4 offer the first possibility.

The combination of quantitative and qualitative methods are semiqualitative ones. The semiqualitative simulation gives quantitative and qualitative information of the predicted states of the system. It combines advantages of the qualitative simulation (ability to work with limited information) and advantages of the quantitative simulation (quantitative prediction) [14].

The present tendency is the increasing use of the quantitative information in qualitative simulation. In fact, there are some simulators that are qualified as semiqualitative and it is not clear which are their semiqualitative aspects, hence it could be better to call them quantitative simulators [23].

Some of the existing semiqualitative simulators based on qualitative simulators are described in this section.

3.5.1 Q2

Q2 [55] is an extension of QSIM [47]. Its name derives from the fact that it is a qualitative and quantitative simulator. The basis of this simulator is QSIM to which an interval arithmetic [68] module has been added. This module works in parallel with QSIM and computes the value of each variable using the numeric constraints. The final value is computed by intersecting the values given by this module and by QSIM [49]. Therefore, the value of each variable is no more qualitative (a predefined label). It is an interval value, whose ends are not predefined [23].

Sometimes the intersection reduces the range of the variable and sometimes the result of the intersection is an empty set. The second possibility means that the state predicted by the qualitative simulator is impossible so it must be eliminated. In conclusion, the envelopes given by the qualitative simulator are tightened through a filter. These new envelopes are less overbounded than the ones obtained using only qualitative simulation.

Simulators that use numeric information for filtering, like Q2, are enhancements with respect to qualitative simulators. However, this is not a big enhancement because they do not take a great advantage of the quantitative knowledge of the system, hence the envelopes could be tighter.

The values of the time points can be calculated using first order Taylor-Lagrange formula. However, it is proved in [63] that this method does not provide relevant information in the neighbourhood of critical points with zero derivatives [89]. Unfortunately, most of the QSIM state events correspond to such points.

3.5.2 Q3

Many semiqualitative simulators, like Q2, are divided into two parts: the qualitative one and the quantitative one. These two parts are not separated in Q3 [14] so it can be seen as a qualitative simulator that gives numeric information or as a quantitative simulator that gives qualitative information.

An interval is a way to represent information that links conceptually quantitative and qualitative representations. The simulator Q3 uses intervals to represent partial information in a flexible way and interval arithmetic for constraint propagation.

Some simulators like QSIM obtain a series of qualitatively different states. The time points in which there are changes from one state to another are not indicated or they are indicated imprecisely (giving a time interval instead of a time point, for instance). The outputs of the quantitative simulators are more precise if the simulation time step decreases. This principle is applied by Q3 to the qualitative simulation. It inserts intermediate states in the qualitative simulation hence the number of states increases, the number of constraints increases and the precision increases too. It is convergent: if more intermediate states are added, the results are more precise.

It is also a stable simulator, because when the precision of the initial conditions increases, the precision of the results increases too. This is due to the use of the Waltz filter for the constraints. A constraint to be used for the Waltz filter can not require widening an interval when another one tightens.

The intervals obtained would be correct if the operations performed by the computer were correct. Nevertheless, the computers work with digital numbers instead of working with real numbers and there are rounding errors. Hence, Q3 adds, if it computes the higher end of an interval, or subtracts, if it computes the lower end of an interval, 10^{-6} at each operation. Another problem is the computing effort needed, which increases when the number of intermediate states increases. The importance of these two problems increases faster than precision.

3.5.3 SQSIM

SQSIM (Semiqualitative Simulator) [49] is a combination of several simulators: QSIM, Q2, Q3 and NSIM. It intersects the results given by each of these simulators. If various overbounded envelopes are intersected, a new tighter overbounded envelope is obtained. If any of the envelopes intersected is not overbounded, the result will be tighter, but it will not be overbounded. This is the problem of SQSIM: some of the simulators used, give envelopes with unknown properties. Hence the envelopes obtained using SQSIM are tighter but have unknown properties as well.

3.5.4 Ca∼En

Ca∼En [17] is a model-based supervision system for on-line applications. It includes a semi-qualitative simulator that is based on causal and constraint reasoning. It uses an explicit representation of time given by a logical clock. Its frequency is adjusted depending on the characteristics of the process. It is used to synchronise the simulator with the measurements of the process and to evaluate the duration of the states and the dates of the events.

The value of a variable is an interval of the real axis. The models are represented using two levels: a local constraint level and a global constraint level. Both levels can manage imprecise knowledge.

In the local constraint level, the influences between the variables are made explicit by the use of an oriented graph. An influence is a causal relation between two variables. For instance: if A increases then B increases too, hence A influences B. There are four types of influences: dynamic, integral, static and constant. The dynamic influence is supposed to be a first order one. Therefore, some parameters must be indicated: static gain, time constant, delay, etc. A condition to activate the relation can be specified. If the relation is not a first order one or it is not linear, it must be linearised and approximated by a first order relation. The internal representation of influence relations is in the form of discrete time equations.

On the other hand, in the global constraint level, constraints derived from physical laws are implemented. They are functions relating numeric variables.

Therefore, Ca∼En uses a multi-model knowledge representation [22].

The simulation algorithm is synchronous and includes two phases. The first one propagates the values of the variables, given as numeric intervals, through the oriented graph, using interval arithmetic and optimisation. As the relations are linear in the local constraint level, the values of the variables are updated applying the superposition principle. The updated value of a variable is obtained as a weighted sum (attending the influences) of the values of all the variables that influence it. The second phase uses the global constraint model to refine the values obtained from the first phase. They must be refined due to the use of interval arithmetic and due to linearisations. The output of the system is the trajectory of each process variable represented by an envelope.

In a first version of Ca$\sim$En the value spaces for the variables and their derivatives were predefined. Hence, they could be considered as qualitative values. Later versions eliminated these value spaces and now the value of each variable can be any interval.

Although the one step ahead prediction is optimised in order to take the multi-occurrence of the variables and their dependencies into account, the obtained envelopes are overbounded due to ignored dependencies across time, i.e. the systems are considered time variant.

3.6 Semiqualitative simulation based on qualitative simulation and fuzzy logic

This section presents a description of some semiqualitative simulators based on fuzzy logic. It will be seen that all of them end up using interval arithmetic, like the simulators described in section 3.5, after fuzzy sets are converted in intervals through α-cut, which can be seen in figure 3-6.

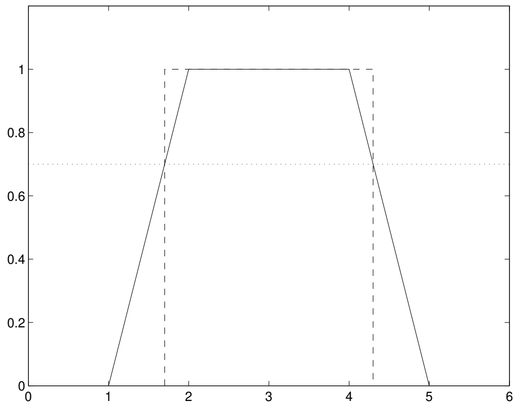


Figure 3-6: Interval obtained by α-cut at 0.7 of a fuzzy number.

3.6.1 FuSim

FuSim (Fuzzy Qualitative Simulator) [76] is an extension of QSIM. It is not constructive, like QSIM. In FuSim, variables' values are given by intervals instead of qualitative labels. It is based on the same two phases than QSIM: qualitative generation of all possible transitions and filtering, using numeric information, of the transitions that do not fulfil the constraints. The mean value theorem is used for the filtering phase.

The only time information that gives QSIM is the order of the future states. FuSim indicates the time that the system will remain in each state [23]. It can generate this information because there is a difference with respect to QSIM. Like QSIM, FuSim represents the variables with their value and the value of their derivative, but the derivative can have a fuzzy value so it is not limited to be positive, negative or zero. This allows, for instance, comparing the predictions of FuSim with the measurements of the variables. This time is computed using the first order Taylor-Lagrange formula (see section 3.5.1), like Q2 does, and hence it has the same disadvantages than Q2 near the points with zero derivatives.

FuSim converts the fuzzy numbers to intervals using α-cut. However, the intervals obtained are of a special kind because there is a limited number of possible intervals. This number is fixed when the value space for each variable is defined.

At each time point, the value of each variable is a fuzzy number belonging to the value space of that variable. This value space is predefined discretising the real axis, that is to say splitting the real axis in a finite number of subsets. The predicted values of the variables do not coincide, in general, with the possible fuzzy values in the quantity space of the variables. Then the algorithm chooses the likeliest value using the metric distance (a measure of the similarity of two fuzzy numbers) and the state priorisation (a single variable can not have different values for different constraints). This action makes FuSim unsound (it includes spurious states) and incomplete (all possible states are not included).

3.6.2 Mycroft

Mycroft [23] includes two simulators (a non-constructive simulator and a semi-constructive one) plus other elements. These simulators are centred in dynamic continuous systems, linear or not, and they are based on QSIM, FuSim and PA. Mycroft takes advantage of the best characteristics of each one and enhances some of them. For instance, some characteristics inherited from these simulators are:

- It works with n derivatives of each variable, like PA. QSIM and FuSim work only with the first derivative.
- The value of the variables is expressed with fuzzy sets, hence it generates time information. This is a characteristic that FuSim and the quantitative simulators have and QSIM and PA do not have.
- The equations are causally ordered, hence the simulation is more constructive. PA is constructive, but QSIM and FuSim are not constructive.

Some of the new features are:

- The way to deal with intermediate variables.
- Only the future values of the state variables are computed. The three simulators mentioned above compute the future values of all variables. It is not necessary to do it if the equations are causally ordered.
- A new priorisation algorithm. This is an enhancement with respect to FuSim.
- The problem of the prediction of impossible behaviours is partially solved because the intermediate variables have interval values and they do not have a predefined value space (see section 3.4.1). To do this, the fuzzy numbers are converted to intervals using α-cut and interval arithmetic is used to compute the predicted states. The computed results still include impossible states due to the use of interval arithmetic (monotonic inclusion, multi-incidences, etc.) but they do no longer include impossible values due to the use of value spaces.

The semi-constructive simulator uses the transition rules for the generation of possible new values and for filtering and erasing impossible values as well. It is not complete. A proof is that in some simulations there are not predicted future states because all the generated transitions are filtered out.

Both simulators (the constructive one and the semi-constructive one) are not envelope generators because the result of the simulation is a tree that indicates the successive states of the system. The time information that is generated is used only to determine the priority between the different transitions. It does not inform the user when these transitions will occur.

3.6.3 Qua.Si.

Qua.Si. [15, 16] are three simulators (Qua.Si. I, II and III) for continuous dynamic systems. The system model is given in the form of differential equations in which parameters or initial conditions are uncertain and can take fuzzy values. The goal is to extend the traditional quantitative simulators by means of Fuzzy Sets Theory, hence considering the mathematics of the real numbers as a particular case of fuzzy mathematics. Therefore, if the model is quantitative the results must be the same than the ones that would be obtained with a quantitative simulator.

One way to solve the differential equations could be converting the fuzzy numbers to intervals, by means of α-cut, and applying interval arithmetic. The results will not be good because interval arithmetic considers that each incidence of a single variable is independent one from another.

The description of the problem follows. At the initial state, there is an uncertainty region in the state space. The shape of this region is an hypercube and it is desired to know its evolution across the time. It is not necessary to study the evolution of every point that belongs to the initial hypercube. Under general conditions of continuity and differentiability, it is necessary to study only the evolution of all the points that belong to the hypercube surface. The number of trajectories to be studied still is infinite, although it is of a smaller order of magnitude.

One way to solve this problem is transforming a sampling problem into an optimisation problem. The goal is finding the maximum and the minimum values for each variable at each time step, taking into account that the initial value must be in one of the sides of the hypercube.

Qua.Si. I

Qua.Si. I uses a scalar systems method, described in section 3.2.2, reconstructing the hypercube at every time step. It simulates characteristic points of the hypercube in the state space: the vertices, the centre of the edges, the centre of the faces, etc. Therefore, it produces incomplete envelopes. Moreover, the envelopes are unsound, due to the wrapping problem, and unstable.

The model of the system can be described with algebraic and differential equations. There is a possibility to do it with fuzzy rules as well.

Qua.Si. II

In Qua.Si. II, the hypercube is not reconstructed at every time step hence avoiding the wrapping problem. Therefore, external behaviours are not added and envelopes are underbounded, as was proven in section 3.2.2.

Qua.Si. III

Qua.Si. I and Qua.Si. II use heuristic criteria to select the trajectories. Qua.Si. III treats the propagation of the fuzzy distributions across the phase space as an optimisation problem with multivariable constraints. This allows the use of advanced numerical techniques whose results are similar, in precision, to the ones obtained using optimisation methods. To enhance the performance of the optimisation algorithm, the function to be optimised and its partial derivatives are used.

The optima that it finds are not guaranteed because it uses optimisation methods with no guarantee. The computational complexity grows exponentially with time because at each time point the optima are computed with respect to the initial state. It was demonstrated in section 3.2.2 that the maximum at one time step and the maximum at the previous one perhaps do not belong to the same trajectory. Hence, all the trajectories starting at the initial state must be computed to find the optima. The computational complexity also grows exponentially with the order of the system because the initial hypercube has more surfaces.

Conclusions

According to the description made of these three simulators some conclusions can be extracted:

- Qua.Si. I. The envelopes obtained can be unstable. They are unsound due to the addition to the hypercube of states that do not belong to it. Moreover, they can be incomplete because it simulates only quantitative models. Therefore, they are neither sound nor complete.

- Qua.Si. II. The envelopes are no longer unsound, but they can still be incomplete. Hence, they are underbounded.

- Qua.Si. III. The simulation problem is translated into an optimisation problem. The obtained envelopes can be incomplete and hence underbounded because the optimisation algorithm used can provide local optima.

These simulators are based on interval arithmetic, although uncertainties are given by fuzzy distributions, because these distributions are converted into intervals by means of α-cut in order to perform mathematical operations.

3.6.4 FRenSi

FRenSi (Fuzzy Region Simulator) [53] is a simulator based in the quantitative models method (see section 3.2.2) hence its results are underbounded. The parameters of the model can be represented by fuzzy sets. FRenSi also deals with these parameters by using the α-cut.

3.7 Summary

Different options to simulate the behaviour of uncertain systems have been presented in this chapter. The existing simulators can be classified into three groups:

- Quantitative. Numerical techniques are applied to deal with uncertainty expressed with intervals:

  - Fixed threshold calculus. The completeness of the envelopes is guaranteed only if the threshold is high, but this is not very useful because then the envelopes are much overbounded.

  - Adaptive threshold calculus using stochastic methods. These methods are useful when the uncertainty can be expressed with probability distributions. Their results are envelopes with a certain degree of confidence. They are not adequate for interval systems.

  - Simulation of quantitative models. The obtained envelopes are underbounded.

  - Simulation as a range determination problem. If an optimisation method is used and there is no guarantee that it will find the global optima, the envelopes will be underbounded.

- Qualitative. They are based on qualitative reasoning. These simulators are designed to deal with qualitative information, not with quantitative information like the information provided by an interval model. Hence, they do not take advantage of this numerical information. Therefore, the original QSIM brings overbounded envelopes, but there are some optional filters that make it give incomplete envelopes. On the other hand, PA is incomplete because it finds only one possible behaviour, not all of them.

- Semiqualitative. There are two types of semiqualitative simulators:

  - Based on qualitative reasoning plus numerical knowledge. Q2 and Q3 obtain tighter envelopes than QSIM adding numerical techniques (interval arithmetic) to it. The properties of the envelopes (completeness, soundness) have not been studied. SQSIM intersects envelopes produced by QSIM, Q2, Q3 and NSIM. If all the envelopes generated by these simulators were overbounded, the intersection will be overbounded too, but tighter. However, as it was stated above, some of these simulators have unknown properties. Hence, the envelopes are tighter but their properties are unknown. Ca$\sim$En provides overbounded envelopes. The semiqualitative simulators based on fuzzy logic FuSim and Mycroft can be included in this group because, as it has been seen, they also use interval arithmetic. FuSim gives unsound and incomplete envelopes. The envelopes provided by Mycroft are incomplete.
  - Not based on qualitative reasoning. The interval methods can be combined with some numerical methods like the adaptive threshold calculus or the optimisation thus giving semiqualitative methods. The interval methods guarantee the results because the natural extension of a function has the monotonic inclusion property. The envelopes obtained are overbounded, much overbounded. This is the case of NSIM and NIS. The properties of the simulator of Gasca *et al* have not been studied yet. Some simulators based on fuzzy logic can be included in this group. All the Qua.Si. simulators generate incomplete envelopes. Qua.Si. I and Qua.Si. II because

they are based on the quantitative models method, like FRenSi. Moreover, the envelopes of Qua.Si. I are unsound. The envelopes of Qua.Si. III also are incomplete because it uses optimisation methods without guarantee.

These conclusions are summarised in table 3.1. When the resulting envelopes are neither complete nor sound they can not be considered neither as overbounded nor as underbounded. This does not mean, unfortunately, that the envelopes are exact.

Type		Simulator		Overbounded	Neither complete nor sound	Underbounded
Quantitative		Fixed threshold		*	*	*
		Adaptive threshold		*	*	*
		Quantitative models				√
		Optimisation		*	*	*
Semiqualitative	quantitative + interval	Moore		√		
		NSIM		√		
		NIS			√	
		Gasca			?	
	qualitative + interval	Q2		?		
		Q3		√		
		SQSIM		?		
		Ca∼En		√		
	qualitative + fuzzy	FuSim			√	
		Mycroft				?
		Qua.Si.	I		√	
			II			√
			III			√
		FRenSi				√
Qualitative		QSIM	original	√		
			opt. filters		?	
		PA				?

Table 3.1: Summary of the properties of the simulators.

The properties of the envelopes obtained by means of some methods depend on their particular application. For instance, a simulator based on the range determination method will provide overbounded or underbounded envelopes depending on the particular algorithm used. If it is an optimisation algorithm and finds inner approximations to the optima, the envelopes are underbounded and if this algorithm gives outer approximations the envelopes are overbounded. These methods are indicated by *.

In conclusion, there exist many simulators for models in which uncertainty is represented. They can generate underbounded or overbounded envelopes or envelopes which are neither complete nor sound. In many cases the authors of the simulator do not assess the properties of the envelopes. In some of these cases the properties can be deduced studying the algorithms used to simulate, but in other occasions the properties remain unknown. In these cases, the properties are indicated by ? if there are doubts about them.

Chapter 4

Generation of envelopes

4.1 Introduction

One of the techniques to generate envelopes is the conversion of the simulation problem into a problem of determination of the range of a function into a parameter space. The properties of the obtained envelopes will depend on the properties of the obtained approximation to the range.

This chapter presents a simulator for interval models based on this technique but in a new approach: the use of MIA, which is also introduced. The main goal for presenting this simulator is to have a deeper knowledge of this method. This will show its advantages and disadvantages and will allow to define the characteristics of a new simulation method.

4.2 Global optimisation methods

The goal of a global optimisation method is to find the maximum and the minimum of a function. That is finding the maximum (resp. the minimum) value of the function and its location, i.e. the combination (or combinations if there are several maxima) of parameters that maximise (resp. minimise) the function. There can be constraints, for instance when the function is optimised in a parameter space, or not. They are called global to distinguish them from the optimisation methods that in some cases give local optima instead of the global ones.

In the case of the computation of the range of a function in a parameter space, it is necessary to determine the optima of the function, but it is not necessary to determine where these optima are located.

There are many methods for global optimisation [50]. Global optimisation methods based on interval arithmetic [36] obtain overbounded results due to the monotonic inclusion property of the interval extensions of functions. When the range of a function f in a parameter space A is computed by means of its interval natural extension $fR(A)$, the obtained result is an overbounded approximation of the exact range $f(A)$:

$$fR(A) \supseteq f(A) \tag{4.1}$$

The degree of overbounding depends on the width of the input intervals. The tighter are the inputs, the tighter is the result, but always overbounded. Therefore, a way to obtain less overbounded results is splitting the parameter space into subspaces

$$A = A_1 \cup A_2 \cup \ldots \cup A_n \tag{4.2}$$

and computing the range of the function at each subspace. The union of these ranges is a less overbounded approximation of $f(A)$:

$$fR(A) \supseteq (fR(A_1) \cup fR(A_2) \cup \ldots \cup fR(A_n)) \supseteq f(A) \tag{4.3}$$

Therefore, successive better approximations are obtained when the parameter space is split in subspaces smaller and smaller, but it is not necessary to split some of the subspaces. This is what branch and bound algorithms do.

One of the drawbacks of interval arithmetic when used for this task is that it does not take into account that a variable appearing several times into an expression is a single variable. Similarly, it is unable to take into account dependencies of other types: interdependent variables (there is a relation among them) or variables that depend of the same variable. This drawback is overcome in some cases by MIA, which is introduced in the next section. In addition to the treatment of multi-incidences, MIA has some interesting properties for the work presented in this thesis. For instance, it can be used not only to obtain overbounded approximations of the range of functions. In addition to them, underbounded approximations can be obtained as well.

4.3 Introduction to Modal Interval Analysis

4.3.1 Definitions and properties

MIA [31, 29, 30] extends real numbers to intervals. Unlike classical interval analysis which identifies an interval by a set of real numbers, MIA identifies the intervals by the set of predicates that are fulfilled by the real numbers.

In the following, some of the properties of modal intervals that are interesting for envelope generation are stated. The proofs of all the results presented in this section as well as other recent results of modal intervals can be found in [77].

Given the set of closed intervals of $\mathbb{R}$, $I(\mathbb{R}) = \{[a,b]' \mid a,b \in \mathbb{R}, \quad a \leq b\}$, and the set of logical existential and universal quantifiers $\{E, U\}$, a modal interval is defined by a pair:

$$X := (X', QX) \tag{4.4}$$

in which $X' \in I(\mathbb{R})$ and $QX \in \{E, U\}$. X' is called the *extension* and QX is the *modality*. In a similar way that real numbers are associated in pairs having the same absolute value but opposite signs, the modal intervals are associated in pairs too, each member corresponding to the same closed interval of the real line but having each one of the opposite selection modalities, existential or universal.

The universal and existential quantifiers are represented by U and E, using a more general notation than the classic $\forall$ and $\exists$. Moreover, since the quantifiers are operators which transform real predicates into interval predicates, it will be written $E(x, A')\,P(x)$ and $U(x, A')\,P(x)$, indicating both arguments, the real index x and the interval argument A'. The notation $E(x, A')$ $P(x)$ is preferred to the more verbalistic and intuitive form $(\exists x \in X')\ P(x)$. Indeed, when the second argument is not a set, like in the following definition, the later notation is outwardly misleading.

The canonical notation for modal intervals is:

$$[a_1, a_2] := \left\{ \begin{array}{lll} \left([a_1, a_2]', E\right) & \text{if} & a_1 \leq a_2 \\ \left([a_2, a_1]', U\right) & \text{if} & a_1 \geq a_2 \end{array} \right\} \tag{4.5}$$

A modal interval $\left([a_1, a_2]', E\right)$ is called "existential interval" or "proper interval" whereas $\left([a_2, a_1]', U\right)$ is called "universal interval" or "improper interval".

4.3.2 Modal interval relations and operations

The set of modal intervals is denoted by $I^*(I\!R)$. The modal quantifier Q associates to every real predicate $P(.) \in \text{Pred}()$ a unique interval predicate: for a variable x on $\mathbf{R}$ and a modal interval $\left(A', QA\right) \in I^*(I\!R)$,

$$Q\left(x, \left(A', QA\right)\right) := QA\left(x, A'\right) \tag{4.6}$$

Defining the set of real predicates accepted by a modal interval

$$\text{Pred}\left(\left(A', QA\right)\right) := \left\{P(.) \in \text{Pred}(I\!R) \mid \quad Q\left(x, \left(A', QA\right)\right) P(x)\right\} \tag{4.7}$$

the equivalent of the inclusion of classical intervals can be introduced into the system of modal intervals: for $A, B \in I^*(I\!R)$

$$A \subseteq B \Longleftrightarrow \text{Pred}(A) \subseteq \text{Pred}(B) \tag{4.8}$$

In terms of the canonical notation, the inclusion is characterised by

$$[a_1, a_2] \subseteq [b_1, b_2] \Longleftrightarrow (a_1 \geq b_1,\ a_2 \leq b_2) \tag{4.9}$$

In a dual way, it is possible to define the set of predicates rejected by a modal interval

$$\text{Copred}\left(\left(A', QA\right)\right) := \left\{P(.) \in \text{Copred}(I\!R) \mid \quad \neg Q\left(x, \left(A', QA\right)\right) P(x)\right\} \tag{4.10}$$

Pred and Copred are complementary by means of the Dual operator:

$$\mathrm{Dual}\left([a_1, a_2]\right) = [a_2, a_1] \tag{4.11}$$

and

$$A \subseteq B \Longleftrightarrow \mathrm{Dual}\left(A\right) \supseteq \mathrm{Dual}\left(B\right) \Longleftrightarrow \mathrm{Copred}\left(A\right) \supseteq \mathrm{Copred}\left(B\right) \tag{4.12}$$

An aspect to be taken into consideration is the rounding of computations. Computers work with digital numbers, not with real numbers. In order to maintain the relations, direct roundings (up or down) have to be used. If $DI \subseteq I\!R$ is a digital scale for real numbers, then the set of modal digital intervals is $I^*\left(DI\right) = \{[a,b] \in I^*\left(I\!R\right) \mid a, b \in DI\}$ and the modal outer and inner roundings of $A \in I^*\left(I\!R\right)$ are defined by:

$$\mathrm{Inn}\left([a,b]\right) = \left[\mathrm{Right}\left(a\right), \mathrm{Left}\left(b\right)\right] \in I^*\left(DI\right) \tag{4.13}$$

$$\mathrm{Out}\left([a,b]\right) = \left[\mathrm{Left}\left(a\right), \mathrm{Right}\left(b\right)\right] \in I^*\left(DI\right) \tag{4.14}$$

The condition

$$\mathrm{Inn}\left([a,b]\right) \subseteq [a,b] \subseteq \mathrm{Out}\left([a,b]\right) \tag{4.15}$$

is fulfilled and the equality

$$\mathrm{Inn}\left(A\right) = \mathrm{Dual}\left(\mathrm{Out}\left(\mathrm{Dual}\left(A\right)\right)\right) \tag{4.16}$$

makes unnecessary the implementation of the inner rounding.

The structure $\left(I^*\left(I\!R\right), \subseteq\right)$ is a lattice and the minimum and the maximum for a family of n modal intervals $\{A\left(i\right) \mid A\left(i\right) \in I^*\left(I\!R\right)\ ,\ i \in I = (1, \ldots, n)\}$ are called *meet* and *join*. In terms of the bounds of $A\left(i\right) = \left[a_1\left(i\right), a_2\left(i\right)\right]$ they become:

- Meet:

$$\wedge\left(i, I\right) A\left(i\right) = \left[\max\left(i, I\right) a_1\left(i\right), \min\left(i, I\right) a_2\left(i\right)\right] \tag{4.17}$$

- Join:

$$\vee\left(i, I\right) A\left(i\right) = \left[\min\left(i, I\right) a_1\left(i\right), \max\left(i, I\right) a_2\left(i\right)\right] \tag{4.18}$$

Generalising, the set of n-dimensional intervals is

$$I^*(I\!R^n) := \{([a_1,b_1],\ldots,[a_n,b_n]) \mid [a_1,b_1] \in I^*(I\!R),\ldots,[a_n,b_n] \in I^*(I\!R)\} \tag{4.19}$$

and, given $A = (A_1,\ldots,A_n)$ and $B = (B_1,\ldots,B_n)$, the inclusion becomes

$$A \subseteq B \Longleftrightarrow A_1 \subseteq B_1,\ldots,A_n \subseteq B_n \tag{4.20}$$

4.3.3 Semantic extensions of continuous real functions

The dual formulation of the modal intervals allows one to define two "semantic" interval functions, noted by f^* and f^{**} respectively. These play a very important role in the theory because they are in close relation with the modal interval extensions and provide meaning to the interval computations.

Definition 8 (* and **-semantic functions) *If f is an $I\!R^n$ to $I\!R$ continuous function and $A \in I^*(I\!R^n)$ then*

$$\begin{aligned} f^*(A) &:= \vee\left(a_p, A_p'\right) \wedge \left(a_i, A_i'\right) [f(a_p,a_i), f(a_p,a_i)] \\ &= \left[\min\left(a_p, A_p'\right) \max\left(a_i, A_i'\right) f(a_p,a_i), \max\left(a_p, A_p'\right) \min\left(a_i, A_i'\right) f(a_p,a_i)\right] \end{aligned} \tag{4.21}$$

$$\begin{aligned} f^{**}(A) &:= \wedge\left(a_i, A_i'\right) \vee \left(a_p, A_p'\right) [f(a_p,a_i), f(a_p,a_i)] \\ &= \left[\max\left(a_i, A_i'\right) \min\left(a_p, A_p'\right) f(a_p,a_i), \min\left(a_i, A_i'\right) \max\left(a_p, A_p'\right) f(a_p,a_i)\right] \end{aligned} \tag{4.22}$$

where $[f(a_p,a_i), f(a_p,a_i)]$ is a point interval ($a_1 = a_2$) and (a_p,a_i) is the component splitting corresponding to $A = (A_p, A_i)$, with A_p a subvector containing the proper components of A and A_i a subvector containing the improper components of A.

Using this definition, the expressions of the arithmetic operations by means of the interval bounds can be obtained. As an example, this definition is applied in the following to the operation addition.

Being f the function $a+b$ and given $A_1 = [\underline{a_1}, \overline{a_1}]$ and $A_2 = [\underline{a_2}, \overline{a_2}]$, the semantic extensions f^* and f^{**} will be:

1. A_1 and A_2 are proper:

$$\begin{aligned} f^*(A) &= [\min(a_1)\min(a_2)(a_1+a_2), \max(a_1)\max(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.23}$$

$$\begin{aligned} f^{**}(A) &= [\min(a_1)\min(a_2)(a_1+a_2), \max(a_1)\max(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.24}$$

2. A_1 is proper and A_2 is improper:

$$\begin{aligned} f^*(A) &= [\min(a_1)\max(a_2)(a_1+a_2), \max(a_1)\min(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.25}$$

$$\begin{aligned} f^{**}(A) &= [\max(a_2)\min(a_1)(a_1+a_2), \min(a_2)\max(a_1)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.26}$$

3. A_1 is improper and A_2 is proper:

$$\begin{aligned} f^*(A) &= [\min(a_2)\max(a_1)(a_1+a_2), \max(a_2)\min(a_1)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.27}$$

$$\begin{aligned} f^{**}(A) &= [\max(a_1)\min(a_2)(a_1+a_2), \min(a_1)\max(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.28}$$

4. A_1 and A_2 are improper:

$$\begin{aligned} f^*(A) &= [\max(a_1)\max(a_2)(a_1+a_2), \min(a_1)\min(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.29}$$

$$\begin{aligned} f^{**}(A) &= [\max(a_1)\max(a_2)(a_1+a_2), \min(a_1)\min(a_2)(a_1+a_2)] \\ &= [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \end{aligned} \tag{4.30}$$

In this case $f^*(A) = f^{**}(A) = [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}]$ for any modality of A_1 and A_2 and hence the sum of two intervals is noted $A_1 + A_2$ and in terms of the bounds of A_i becomes

$$A_1 + A_2 = [\underline{a_1}+\underline{a_2}, \overline{a_1}+\overline{a_2}] \tag{4.31}$$

In a similar way, the definition can be applied to the other arithmetic operations and then the rules for their modal interval extensions are obtained:

- Interval to the power of a real number:

$$A^n = \begin{cases} [1,1] & \text{if } n = 0 \\ [\underline{a}^n,\ \overline{a}^n] & \text{if } n \text{ is odd} \\ \left.\begin{array}{ll} [\underline{a}^n,\ \overline{a}^n] & \text{if } \underline{a} \geq 0 \text{ and } \overline{a} \geq 0 \\ [\overline{a}^n,\ \underline{a}^n] & \text{if } \underline{a} < 0 \text{ and } \overline{a} < 0 \\ [0, \max(\underline{a}^n,\ \overline{a}^n)] & \text{if } \underline{a} < 0 \text{ and } \overline{a} \geq 0 \\ [\max(\underline{a}^n,\ \overline{a}^n), 0] & \text{if } \underline{a} \geq 0 \text{ and } \overline{a} < 0 \end{array}\right\} & \text{if } n \text{ is even} \end{cases} \tag{4.32}$$

- Difference:

$$A_1 - A_2 = [\underline{a_1}-\overline{a_2}, \overline{a_1}-\underline{a_2}] \tag{4.33}$$

- Product:

$$A_1 * A_2 = \begin{cases} \left[\underline{a_1 a_2}, \overline{a_1 a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} \geq 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\underline{a_1 a_2}, \underline{a_1}\overline{a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} \geq 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} < 0 \\ \left[\overline{a_1}\underline{a_2}, \overline{a_1 a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} \geq 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\overline{a_1}\underline{a_2}, \underline{a_1}\overline{a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} \geq 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} < 0 \\ \left[\underline{a_1 a_2}, \overline{a_1}\underline{a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} < 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\max\left(\underline{a_1 a_2}, \overline{a_1 a_2}\right), \min\left(\overline{a_1}\underline{a_2}, \underline{a_1}\overline{a_2}\right)\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} < 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} < 0 \\ [0, 0] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} < 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\overline{a_1 a_2}, \underline{a_1}\overline{a_2}\right] & \text{if } \underline{a_1} \geq 0,\ \overline{a_1} < 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} < 0 \\ \left[\underline{a_1}\overline{a_2}, \overline{a_1 a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} \geq 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} \geq 0 \\ [0, 0] & \text{if } \underline{a_1} < 0,\ \overline{a_1} \geq 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} < 0 \\ \left[\min\left(\overline{a_1}\underline{a_2}, \underline{a_1}\overline{a_2}\right), \max\left(\underline{a_1 a_2}, \overline{a_1 a_2}\right)\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} \geq 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\overline{a_1}\underline{a_2}, \underline{a_1 a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} \geq 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} < 0 \\ \left[\underline{a_1}\overline{a_2}, \overline{a_1}\underline{a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} < 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\overline{a_1 a_2}, \overline{a_1}\underline{a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} < 0,\ \underline{a_2} \geq 0 \text{ and } \overline{a_2} < 0 \\ \left[\underline{a_1}\overline{a_2}, \underline{a_1 a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} < 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} \geq 0 \\ \left[\overline{a_1 a_2}, \underline{a_1 a_2}\right] & \text{if } \underline{a_1} < 0,\ \overline{a_1} < 0,\ \underline{a_2} < 0 \text{ and } \overline{a_2} < 0 \end{cases} \tag{4.34}$$

- Division:

$$\frac{A_1}{A_2} = A_1 * \frac{1}{A_2} \text{ if } 0 \notin A_2 \tag{4.35}$$

being

$$\frac{1}{A} = \left[\frac{1}{\overline{a}}, \frac{1}{\underline{a}}\right] \text{ if } 0 \notin A \tag{4.36}$$

These rules are similar to the ones used in interval arithmetic (see section 3.3.1) taking into account that now intervals can be improper.

The value of f^* or f^{**} extensions may have, with further thought, no clear meaning concerning the values of f in its domain. Two key theorems reverse this perspective, giving a complete meaning to the interval results f^* and f^{**} and characterizing them as modal interval extensions, previously defined in logical terms.

Theorem 1 (*-Semantic Theorem) *If $A \in I^* (\mathbb{R}^n)$, $f : \mathbb{R}^n \to \mathbb{R}$ is continuous in A' and there exists an interval which is called $F(A) \in I^*(\mathbb{R})$, then*

$$f^*(A) \subseteq F(A) \Longleftrightarrow U\left(a_p, A_p'\right) Q\left(z, F(A)\right) E\left(a_i, A_i'\right)\left(z = f\left(a_p, a_i\right)\right) \tag{4.37}$$

This interpretation can be read: "For all elements belonging to the proper intervals there exists at least one element in the improper intervals that fulfil the function".

Example 5 $[10, 20] + [20, 15] = [30, 35]$ *means*

$$U\left(a, [10, 20]'\right) E\left(f, [30, 35]'\right) E\left(b, [15, 20]'\right)(a + b = f) \tag{4.38}$$

Dual semantics for proper and improper modal intervals are established by the dual semantic theorem.

Theorem 2 (-Semantic Theorem)** *If $A \in I^* (\mathbb{R}^n)$, $f : \mathbb{R}^n \to \mathbb{R}$ is continuous in A' and there exists an interval which is called $F(A) \in I^*(\mathbb{R})$, then*

$$f^{**}(A) \supseteq F(A) \Longleftrightarrow U\left(a_i, A_i'\right) Q\left(z, \mathrm{Dual}\left(F(A)\right)\right) E\left(a_p, A_p'\right)\left(z = f\left(a_p, a_i\right)\right) \tag{4.39}$$

Example 6 *The only semantic interpretation of*

$$[1, 2] + [5, 7] = [6, 9] \tag{4.40}$$

in the context of Interval Arithmetic is

$$U\left(a, [1, 2]'\right) U\left(b, [5, 7]'\right) E\left(f, [6, 9]'\right)(a + b = f) \tag{4.41}$$

In addition to this one, in the context of MIA the semantics

$$U\left(f, [6, 9]'\right) E\left(a, [1, 2]'\right) E\left(b, [5, 7]'\right)(a + b = f) \tag{4.42}$$

is possible, too.

The semantics (4.41) is valid for the *-extension of the function and the semantics (4.42) is valid for the **-extension. In this case, both extensions are equal and hence both interpretations are valid. When all the intervals are proper, it can be said that the *-extension is complete and the **-extension is sound. Therefore, when both extensions are equal the result is exact.

In the following example one of the semantics is lost because, although the two extensions of the operation are the same, the two operands are not independent.

Unfortunately, the computation of the *- and **-extensions is, in general, a difficult challenge and hence the usual procedure is to find overbounded computations of f^* and underbounded computations of f^{**} which maintain the semantic interpretations.

When the continuous function f is a rational function, there exist two modal rational extensions. They are obtained by using the computing program defined by the syntax tree of the expression of the function, in which the real arguments are transformed into interval arguments and the real operators are transformed into their * or **-semantic extensions.

For a general class of operators, for instance the arithmetic operations considered above, both *- and **-semantic extensions are equal. In this case, the function defined by the computational program indicated by the syntax of f is called *modal rational function,* $fR(A)$. In general, $fR(A)$ is not interpretable. The interpretation problem for a modal rational function consists in relating it to the corresponding semantic functions, which have standard meaning defined by the semantic theorems.

There are several theorems relating the modal rational function $fR(A)$ to the modal semantic extensions f^* and f^{**}. The following ones give two * and **-interpretable coercions.

Definition 9 *A component x_i of x is uni-incident in a rational function $f(x)$ if it occupies only one leaf of the syntactical tree of f; otherwise, x_i is multi-incident in $f(x)$.*

Theorem 3 *If in $fR(A)$ all arguments are uni-incident, then*

$$f^*(A) \subseteq fR(A) \subseteq f^{**}(A) \tag{4.43}$$

In particular, if all the components of A are uni-incident and with the same modality,

$$f^*(A) = fR(A) = f^{**}(A) \tag{4.44}$$

Theorem 4 *If in $fR(A)$ there are multi-incident improper components and if AT^* is obtained from A by transforming, for every multi-incident improper component, all incidences but one into their duals, then*

$$f^*(A) \subseteq fR(AT^*) \tag{4.45}$$

If all components of A are proper, then $AT^* = A$ and

$$f^*(A) \subseteq fR(A) \tag{4.46}$$

When these computations are performed using digital numbers, appropriate roundings have to be made:

$$f^*(A) \subseteq \mathrm{Out}(fR(AT^*)) \tag{4.47}$$

Theorem 5 *If in $fR(A)$ there are multi-incident proper components and if AT^{**} is obtained from A by transforming, for every multi-incident proper component, all incidences but one into their duals, then*

$$f^{**}(A) \supseteq fR(AT^{**}) \tag{4.48}$$

If all components of A are improper, then $AT^{**} = A$ and

$$f^{**}(A) \supseteq fR(A) \tag{4.49}$$

In this case, the roundings that have to be performed in order to maintain the semantics are:

$$f^{**}(A) \supseteq \mathrm{Inn}(fR(AT^{**})) \tag{4.50}$$

An interpretable rational interval computation program $fR(A)$ may nevertheless result in a loss of information far more important than the one produced by numerical roundings. Then it is very important to find out criteria to characterise the rational interval functions for which $fR(A)$, with an ideal computation (infinite precision), is such that

$$f^*(A) = fR(A) = f^{**}(A) \tag{4.51}$$

In this case, it is said that $fR(.)$ is optimal for A.

There are several results which characterise the optimality of a modal rational function according to its monotonicity properties.

Definition 10 *A continuous function $f(x,y)$, is an uniformly monotonic function with respect to x in a domain $\left(X^{'},Y^{'}\right) \subseteq (\mathbb{R}, \mathbb{R}^{m})$ if it is a monotonic function with respect to x in $X^{'}$ and keeps the same direction of monotonicity for all the values $y \in Y^{'}$.*

Definition 11 *A continuous function $f(x,y)$, is a totally monotonic function with respect to a multi-incident variable x in a domain $\left(X^{'},Y^{'}\right) \subseteq (\mathbb{R}, \mathbb{R}^{m})$ if it is a uniformly monotonic function with respect to x in $X^{'}$ and, for every incidence of x considered as an independent variable, it is also a uniformly monotonic function.*

Theorem 6 (optimal coercion for uni-modal arguments) *Let A be an interval vector and fR defined in the domain $A^{'}$ and totally monotonic for all its multi-incident components. Let AD be the enlarged vector of A, such that each incidence of every multi-incident component is included in AD as an independent component, but transformed into its dual if the corresponding incidence-point has a monotonicity sense contrary to the global one of the corresponding A-component. Then*

$$f^{*}(A) = fR(AD) = f^{**}(A) \tag{4.52}$$

Theorem 7 (*-partially optimal coercion) *Let A be an interval vector and fR defined in the domain $A^{'}$ and totally monotonic for a subset B of multi-incident components. Let ADT^{*} be the enlarged vector of A, such that each incidence of every multi-incident component of the subset with total monotonicity is included in ADT^{*} as an independent component, but transformed into its dual if the corresponding incidence-point has a monotonicity sense contrary to the global one of the corresponding B-component; for the rest, the multi-incident improper components are transformed into their dual in every incidence except one. Then*

$$f^{*}(A) \subseteq fR(ADT^{*}) \tag{4.53}$$

Theorem 8 (-partially optimal coercion)** *Let A be an interval vector and fR defined in the domain $A^{'}$ and totally monotonic for a subset B of multi-incident components. Let ADT^{**} be the enlarged vector of A, such that each incidence of every multi-incident component* *of the subset with total monotonicity is included in ADT^{**} as an independent component, but transformed into its dual if the corresponding incidence-point has a monotonicity sense contrary to the global one of the corresponding B-component; for the rest, the multi-incident proper components are transformed into their dual in every incidence except one. Then*

$$f^*(A) \supseteq fR(ADT^{**}) \tag{4.54}$$

4.3.4 Examples of range computations

Follow some examples of the application of these theorems to the determination of the range of a function in a parameter space. The first one shows the application of theorem 6 when there are multi-incident variables and the function is optimal.

Example 7 *Determination of the range of $f = x - x$ in the parameter space $X = [1, 2]$. The exact range is $R_f = [0, 0]$ because all possible values of x fulfil $x - x = 0$. The modal rational function*

$$fR(X) = X - X = [\underline{x}, \overline{x}] - [\underline{x}, \overline{x}] = [\underline{x} - \overline{x}, \overline{x} - \underline{x}] = [-1, 1] \tag{4.55}$$

*is an overbounded approximation to $f^*(X)$, according to theorem 4. Therefore it is *-semantically interpreted*

$$U\left(x, [1, 2]'\right) E\left(f, [-1, 1]'\right)(x - x = f) \tag{4.56}$$

The function is monotonic with respect to x

$$\frac{\partial f}{\partial x} = 1 - 1 = 0 \tag{4.57}$$

and with respect to every incidence of x considered as an independent variable

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= 1 > 0 \\ \frac{\partial f}{\partial x_2} &= -1 < 0 \end{aligned} \tag{4.58}$$

and therefore it is totally monotonic and theorem 6 can be applied to obtain the exact range of *the function:*

$$f^*(X) = f^{**}(X) = fR(XD) = X - \mathrm{Dual}(X) = \mathrm{Dual}(X) - X = [0,0] \tag{4.59}$$

*This result is *-semantically interpretable*

$$U\left(x,[1,2]'\right)E\left(f,[0,0]'\right)(x - x = f) \tag{4.60}$$

*and **-semantically interpretable*

$$U\left(f,[0,0]'\right)E\left(x,[1,2]'\right)(x - x = f) \tag{4.61}$$

The following example shows that sometimes it is necessary to use splitting algorithms because the function is not totally monotonic.

Example 8 *Determination of the range of the function $f = x_1x_2 - x_1^2 - 2x_2$ in the parameter space $X_1 = [1,2]$ and $X_2 = [3,4]$. The range of the function is $R_f = [-5,-3.75]$. An overbounded approximation is obtained using the natural extension*

$$f^*(X) \;\subseteq fR(X) = [-9,1] \tag{4.62}$$

The function is monotonic with respect to x_2

$$\frac{\partial f}{\partial x_2} = x_1 - 2 = [-1,0] \leq 0 \tag{4.63}$$

and with respect to every incidence of x_2 considered as an independent variable

$$\begin{gathered}\frac{\partial f}{\partial x_{21}} = x_1 = [1,2] > 0\\ \frac{\partial f}{\partial x_{22}} = -2 < 0\end{gathered} \tag{4.64}$$

hence it is totally monotonic. Nevertheless, it is not totally monotonic with respect to x_1

$$\frac{\partial f}{\partial x_1} = x_2 - 2x_1 = [-1,2] \ni 0 \tag{4.65}$$

Therefore, theorem 7 can be applied to x_2 to obtain a better approximation of the range of the function:

$$f^*(X) \subseteq fR(XDT^*) = X_1 \mathrm{Dual}(X_2) - X_1^2 - 2X_2 = [-8, -1] \tag{4.66}$$

In this case, to obtain even better approximations of the exact range, the parameter space has to be split. The advantage provided by theorem 7 is that it indicates that only the variable x_1 must be split. Moreover, the range in each sub-space can be computed more exactly because theorem 7 has already been applied to x_2. In fact, the function whose range has to be determined now is

$$F = X_1 [4, 3] - X_1^2 - [6, 8]$$

which is an interval function of one variable.

As a conclusion, the number of sub-spaces to be considered to compute an approximation of the range of the function is smaller when modal intervals are used. This is illustrated in [86], where modal intervals combined with a branch and bound algorithm have been applied to the analysis and design of robust controllers.

The next example shows that sometimes it is necessary to compute the range of higher order derivatives in order to compute the range of a function. It has been seen that to study the monotonicity of a function, it is necessary to compute approximations to the range of its derivative. In the examples above, the variables in the derivative are uni-incident so the range obtained using the natural extension is exact. If there are multi-incident variables in the derivative, this range is overbounded.

Example 9 *Determination of the range of the function $f = x_1^2 x_2 - 2x_1^2 + 2x_2^2$ in the parameter space $X_1 = [1, 2]$ and $X_2 = [3, 4]$. The range of the function is $R_f = [19, 40]$. An overbounded approximation is obtained by means of the natural extension*

$$f^*(X) \subseteq fR(X) = [13, 46] \tag{4.67}$$

The function is totally monotonic with respect to x_2, as it is monotonic with respect to x_2:

$$\frac{\partial f}{\partial x_2} = X_1^2 + 4X_2 = [13, 20] > 0 \tag{4.68}$$

and with respect to every incidence of x_2 considered as an independent variable:

$$\frac{\partial f}{\partial x_{21}} = X_1^2 = [1,4] > 0 \tag{4.69}$$
$$\frac{\partial f}{\partial x_{22}} = 4X_2 = [12,16] > 0$$

It seems that the function f is not monotonic when the range of its derivative with respect to x_1 is computed by means of the natural extension:

$$\left(\frac{\partial f}{\partial x_1}\right)^* (X) \subseteq \frac{\partial f}{\partial x_1} R(X) = 2X_1X_2 - 4X_1 = [-2,12] \ni 0 \tag{4.70}$$

However, this is an overbounded approximation of this range, as x_1 is multi-incident. The exact range of the first derivative can be computed studying the monotonicity of the second derivative:

$$\frac{\partial^2 f}{\partial x_1^2} = 2X_2 - 4 = [2,4] > 0 \tag{4.71}$$
$$\frac{\partial^2 f}{\partial x_{11}^2} = 2X_2 = [6,8] > 0$$
$$\frac{\partial^2 f}{\partial x_{12}^2} = -4 < 0$$

Hence, the exact range of the first derivative is:

$$\left(\frac{\partial f}{\partial x}\right)^* (X) = \left(\frac{\partial f}{\partial x}\right)^{**} (X) = \frac{\partial f}{\partial x} R(X) = 2X_1X_2 - 4 \cdot \mathrm{Dual}(X_1) = [2,8] > 0 \tag{4.72}$$

and the function f is totally monotonic:

$$\frac{\partial f}{\partial x_{11}} = 2X_1X_2 = [1,4] > 0 \tag{4.73}$$
$$\frac{\partial f}{\partial x_{12}} = -4 \cdot \mathrm{Dual}(X_1) = [-4,-8] < 0$$

Then, the exact range of f is:

$$f^*(X) = f^{**}(X) = fR(XD) = X_1^2X_2 - 2(\mathrm{Dual}(X_1))^2 + 2X_2^2 = [19,40] \tag{4.74}$$

4.4 An exact envelope simulator for particular cases

To obtain the exact envelope for a time invariant system, it is necessary to determine the range of the functions at each time step taking into account the two types of multi-incidences that appear in these functions.

The first type of multi-incidence is given when a parameter appear several times into a function. Sometimes these multi-incidences do not appear explicitly, for instance if some intermediate operation has been performed or if some parameters depend on another parameter. The second type of multi-incidences is given when the system is assumed to be time invariant and there are parameters that appear in the functions of different time points. These multi-incidences of the second type are converted in multi-incidences of the first type by merging the functions. Then, the range of functions in a parameter space defined by the parameters of the function that are intervals has to be determined.

MIA provides tools to calculate these ranges. These tools are useful in this case as it is necessary to differentiate the function to apply some theorems and the models that are used are difference equations, which are easily differentiable.

A problem appears when the function is not monotonic. In this case, a splitting algorithm has to be used. As the exact envelope is searched for, this algorithm should have to iterate until the exact range is found. In many cases this is not possible because the parameter space can not be split in rectangular monotonic subspaces and moreover there is always an error due to roundings.

A simulator has been implemented. It has been given the name MIS_0 as it is based on MIA and generates the exact envelope (the envelope with error 0) in some special cases.

The characteristics of this simulator are:

- It simulates the behaviour of SISO systems modelled using discrete-time interval equations.
- The system is considered time invariant and hence the envelopes at any time point are determined starting from the origin of the simulation.
- There can be some uncertainty on this origin, hence the initial state can be an interval.

4.4.1 Experimental results

This simulator has been implemented in Maple V release 4 [3], but it can be launched from Matlab 5 [4] by means of the Extended Symbolic Toolbox.

It has been used to simulate the behaviour of a generic first order system

$$y_n = \left(1 - \frac{T}{\tau}\right) y_{n-1} + \frac{kT}{\tau} u_{n-1} \tag{4.75}$$

with the following parameters:

- static gain: $k = [0.95, 1.05]$
- time constant: $\tau = [5, 20]\ s$
- initial state: $Y_0 = [0, 0]$.
- input: u is 1 from 1 to 15 s and 0.5 beyond that point.
- sampling time: $T = 1\ s$.

The results are shown in figure 4-1. There are three envelopes in this figure. They all have the same lower bound, but the higher bound is different. The dashed line one was obtained by simulating the four extreme quantitative models. The dotted line one was obtained making a grid for the two parameters and simulating all the scalar systems belonging to the grid. Finally, the solid line one is the exact envelope obtained by means of MIS_0. It can be seen that the exact envelope contains the envelopes obtained by simulating quantitative models, which are underbounded, and that these envelopes are closer to the exact one when the number of quantitative models increases.

4.5 Summary

This chapter presents the problem of the generation of the exact envelopes. In the general case, this problem can be translated to another problem of range determination. Tools like global optimisation algorithms can be used for this problem. Some of these algorithms are based on Interval Arithmetic, which obtains overbounded results when there are multi-incidences in the

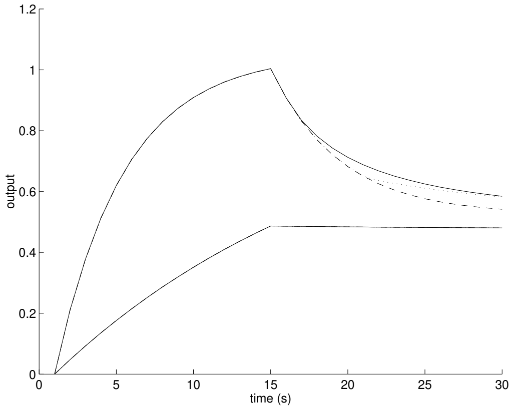


Figure 4-1: Exact envelope.

function at a time point because they are not considered. The results are more overbounded when the systems are considered time invariant, because then another type of multi-incidences appears: multi-incidences between functions at different time points.

Multi-incidences between functions may be converted in multi-incidences in a function combining these functions to form new ones. Then, these multi-incidences in a function can be taken into account using MIA, an extension of Interval Analysis which is introduced in this chapter. The main theorems related to this work, which are applied to some examples, are presented.

A new simulator based on this tool has been developed, MIS_0. It can be used to simulate the behaviour of some particular SISO systems with interval models, inputs or initial states. The output is the exact envelope in these cases. Some examples of simulations are presented.

This simulator has been used to show the problems that appear when the determination of the exact envelopes is intended. As these problems are common to any simulator with this goal, whatever is the method used, the algorithm of MIS_0 is not given. These problems are:

- The exact envelope is not always obtained due to the rounding errors.
- If the exact envelope is obtained, an important computational effort may be needed.
- The procedure is not incremental and therefore the computing effort increases at each step.

In many cases it is not necessary to compute the exact envelopes and hence it is not necessary to make such a computational effort. Next chapter discusses this question and then an appropriate simulator, which is not based on MIS_0, is introduced.

Chapter 5

Generation of error-bounded envelopes

5.1 Introduction

The most important conclusion of the overview in chapter 3 is that all that simulators, independently of the properties of the envelopes that they generate, have a common characteristic: the distance between the obtained envelope and the exact one is not known, i.e. the degree of over or underbounding is unknown.

It is necessary to know the exact envelope to know the distance between an envelope and the exact one and it is assumed that the exact envelope is of course not known, as the exact envelope problem is actually highly complex. The proposed alternative way in order to approach this problem is to bound the error, i.e. to determine the maximum distance. This can be achieved by computing both an underbounded envelope and an overbounded envelope for a given model and a given input. The underbounded envelope is included in the exact one, which is unknown and, in addition, included in the overbounded one. Therefore, these two envelopes bound the exact one and the distance between them indeed gives the maximum error. These envelopes are defined as error-bounded envelopes.

The error between the envelopes can be reduced either widening the underbounded envelope or tightening the overbounded one. Obviously, the computation effort increases when the error decreases.

The used approach to generate the underbounded and overbounded envelopes is based on the reformulation of the simulation problem into a range determination problem and requires the search for inner and outer approximations to the exact range. Both possibilities are achieved by means of MIA.

5.2 Error-bounded envelopes simulation

MIA is used to generate both envelopes as it allows to approximate the two semantic extensions of a function f, $f^{*}(X)$ and $f^{**}(X)$, whose semantic interpretations correspond to those of the overbounded and the underbounded envelopes:

- Overbounded envelope. Its semantics is: "For every (universal quantifier) model parameter, input and initial state, the output belongs to the envelope (existential quantifier)", which corresponds to the *-extension.
- Underbounded envelope. Its semantics is dual: "For every output belonging to the envelope there exist parameter, input and initial state values that produce this output", which corresponds to the **-extension.

A new simulator, MIS, has been created. It is based on a branch and bound algorithm.

5.2.1 Algorithm

This branch and bound algorithm computes an external and an internal approximations to the range of the function at each time point. The algorithm applies the coercion theorems to the monotonic variables, which are determined assessing the monotonicity by using only the first derivative. If the function is not totally monotonic and possibly the maximum or the minimum of the function is in a parameter space, this space is split along the edges of the not monotonic variables. The condition to stop this iterative algorithm is that the error between the two envelopes is lower than the desired one, which is fixed at the beginning of the simulation.

The input arguments of the algorithm are a function f of a set of variables $X = \{x_1 \ldots x_n\}$ and the parameter space determined by the interval values of the variables:

$$Q = \left\{ q = [q_1, q_2, \ldots, q_n]^T \mid q_i \in \left[\underline{q_i}, \overline{q_i}\right], i = 1 \ldots n \right\} \tag{5.1}$$

The output arguments are the external and internal approximations to the range.

---

function $(external, internal) =$ **approx_range**(f, Q)
  $finish = false$
  $internal =$ **inner**(f, Q)
  save$(Q, read\_list)$
  DO
    $external = internal$
    DO
      $P =$ **get**$(read\_list)$
      IF **not_monotonic**$(f, P) = 0$ THEN
        $partial =$ **exact**(f, P)
        $internal = internal \vee partial$
        $external = external \vee partial$
      ELSE
        $partial =$ **inner**(f, P)
        $internal = internal \vee partial$
        $partial =$ **outer**(f, P)
        $external = external \vee \ partial$
        IF **not**$(partial \subseteq internal)$ THEN
          $(P_1, P_2) =$ **split**(P)
          save$(P_1, write\_list)$
          save$(P_2, write\_list)$
        ENDIF
      ENDIF
    WHILE **not**(**is_empty** $(read\_list)$)
    IF **is_empty**$(write\_list)$ THEN
      $finish = true$
    ENDIF
    IF **stop_condition** THEN

$finish = true$

ENDIF

exchange$(read\_list, write\_list)$

WHILE **not**$(finish)$

The functions that are used by the algorithm are described in the following.

function b =**not**(c) returns $true$ if c is $false$ and vice versa.

function save$(Q, list)$ adds the subspace Q to the list of subspaces $list$. The length of the list is increased by one unit.

function Q =**get**$(list)$ picks one of the elements of the list of subspaces $list$. This element is deleted in the list, so the number of elements in the list is decreased by one unit.

function b =**empty**$(list)$ returns $true$ or $false$ depending on whether the list of subspaces $list$ is empty or not, respectively.

function exchange$(list1, list2)$ exchanges the lists of subspaces $list1$ and $list2$.

function a =**not_monotonic**(f, Q) returns the real number a, that is the number of variables with respect to which the function f is not monotonic in the parameter space Q. The test is performed using only the first derivative and using the *-partially optimal coercion theorem.

function Z =**exact**(f, Q) applies the optimal coercion theorem for uni-modal arguments (theorem 6) to have the exact range of the totally monotonic function f in the parameter space Q, which is returned as the interval Z.

function Z =**inner**(f, Q) applies the **-partially optimal coercion theorem (theorem 8) to have an internal approximation of the range of f in the parameter space Q, which is returned as the interval Z.

function Z =**outer**(f, Q) applies the *-partially optimal coercion theorem (theorem 7) to have an external approximation of the range of f in the parameter space Q, which is returned as the interval Z.

function (Q_1, Q_2) =**split**(Q) splits the parameter space Q in two subspaces Q_1 and Q_2 such that $Q_1 \cup Q_2 = Q$ and $Q_1 \cap Q_2 = \emptyset$. Different criteria may be used for the splitting. The implemented function splits along the largest edge among the not monotonic variables.

The function **stop_condition** computes the maximum distance between the intervals $external$ and $internal$, which must fulfil $external \supseteq internal$. The function returns $true$ if it is lower or equal to ε:

---

function b =**stop_condition**($external, internal, \varepsilon$)

IF $\overline{external} - \overline{internal} \leq \varepsilon$ AND $\underline{internal} - \underline{external} \leq \varepsilon$ THEN

$b = true$

ELSE

$b = false$

END

5.2.2 Implementation

This simulation algorithm has been implemented using Matlab version 5.1 for Unix [4]. Symbolic computations are performed using Maple V r4 [3] through the Symbolic Math Toolbox. Moreover, it uses C++ programs as MEX-files to perform modal interval computations with direct rounding and to accelerate the branch and bound algorithm. Direct rounding is necessary to maintain the semantic interpretations when computing by means of intervals, as computers use digital numbers, not real ones.

5.2.3 Example of simulation

This simulator is used to study the behaviour of the generic first order system used in section 4.4.1.

The functions provided to the branch and bound algorithm are the ones of (3.3), which are not linear. Figure 5-1 shows the input applied to this system (steps of different lengths and magnitudes).

With this input it is more difficult to compute the exact envelope because there is no monotonicity, so error-bounded envelopes are determined using MIS. Figure 5-1 also shows the obtained output envelopes (the overbounded one drawn with two solid lines and the underbounded one drawn with two dotted lines) when the error between the envelopes is fixed at a maximum of $\varepsilon \leq 0.2$.

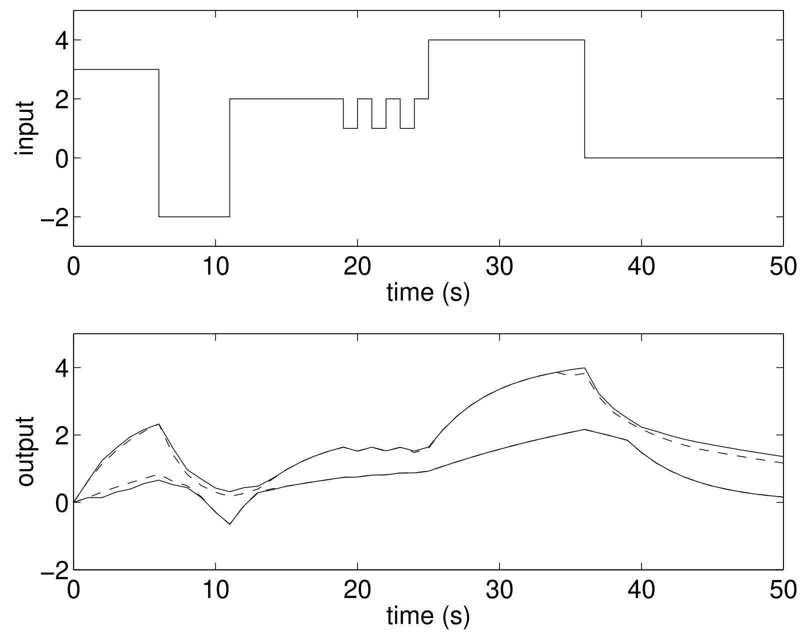


Figure 5-1: Error-bounded envelopes.

5.2.4 Conclusions

MIS provides error-bounded envelopes for an interval model expressed with difference equations and a given interval input. The computation effort needed depends on the error between the two envelopes, which is given to the simulator as a parameter. As an example of the computation effort, in the simulation of the previous section, where the maximum error has been fixed to 0.2, at time point $t = 30$ s the exact envelope is $[1.576211, 3.359412] \subseteq \mathrm{Env}(30) \subseteq [1.576199, 3.359415]$.

The exact envelope at a time point t can only be obtained by computing the range of the function that relates the current time point to the initial one, i.e. all the previous states must be considered in order to obtain the exact envelope. This means that the procedure is not incremental and hence this method needs a computation effort that increases at each step of the simulation, which is a drawback. A pretty good approximation to the computation time has been obtained by means of a polynomial of degree 3.

5.3 Semantics of envelopes using sliding time windows

A possible solution to overcome the problem of the continuous increment in the computation time is the use of sliding temporal windows. Due to the dynamics of the systems, it is known that the influence of the previous states over the current one decreases with time. Hence, it should be possible to obtain close results with a shorter temporal window length. This approach solves the computation effort problem but introduces new problems related to the semantics of the results. This section discusses the problems related to the semantics of the envelopes when sliding time windows are used.

5.3.1 Overbounded envelope

In the case of the overbounded envelope, the semantics associated to the envelope at a time point w, being w the window length, is:

$$\begin{aligned} & U\left(y\left(0\right),Y\left(0\right)\right)U\left(u\left(0\right),U\left(0\right)\right)...U\left(u\left(w-1\right),U\left(w-1\right)\right)U\left(p_i,P_i\right) \\ & E\left(y\left(w\right),Y\left(w\right)\right)\left(y\left(w\right)=f\left(y\left(0\right),u\left(0\right),\ldots,u\left(w-1\right),p_i\right)\right) \end{aligned} \tag{5.2}$$

In this expression, $y\left(0\right)$ is the output of the system at the initial state, $u\left(i\right)$ are the inputs, p_i are the parameters of the system model and $y\left(w\right)$ is the calculated overbounded envelope at the time point w. This expression is read: "for every initial state, input and model parameter, the output belongs to the envelope".

If the envelope at the time point $2w$ is computed using this overbounded envelope at the time point w as its initial state, the semantics of this envelope will be

$$\begin{aligned} & U\left(y\left(w\right),Y\left(w\right)\right)U\left(u\left(w\right),U\left(w\right)\right)...U\left(u\left(2w-1\right),U\left(2w-1\right)\right)U\left(p_i,P_i\right) \\ & E\left(y\left(2w\right),Y\left(2w\right)\right)\left(y\left(2w\right)=f\left(y\left(w\right),u\left(w\right),\ldots,u\left(2w-1\right),p_i\right)\right) \end{aligned} \tag{5.3}$$

It could seem that combining these two expressions the semantics with respect to the initial state will be

$$U\left(y\left(0\right),Y\left(0\right)\right)U\left(u\left(0\right),U\left(0\right)\right)...U\left(u\left(2w-1\right),U\left(2w-1\right)\right)U\left(p_i,P_i\right) \tag{5.4}$$

$$E\left(y\left(2w\right),Y\left(2w\right)\right)\left(y\left(2w\right)=f\left(y\left(0\right),u\left(0\right),\ldots,u\left(2w-1\right),p_i\right)\right)$$

which is the semantics of the desired overbounded envelope. However, there are implicit multi-incidences that have not been taken into account. These multi-incidences are due to the parameters of the system p_i. As stated in section 3.2.3, the parameters of the model appear in the difference equation of every time point and, if the physical system is assumed to be invariant, they can not be treated as independent variables. If they are treated as so, the system is considered time variant and the exact envelopes are wider than the one of the time invariant system. This is what is done when sliding time windows are used. The parameters of the model in the window from 0 to $w-1$ and the parameters from w to $2w-1$ are considered independent. Table 5.1 shows very close approximations to the exact envelopes at time point $t = 50$ s for different window lengths, being the exact envelope of the time invariant system the one computed with a window length of 50 s. The results of this table have been obtained using the generic first order system and applying to it the same input as in section 5.2.3.

window length	overbounded envelope
50	$[0.1583, 1.6357]$
25	$[0.1560, 1.8569]$
10	$[0.1378, 2.2726]$
5	$[0.0932, 2.4580]$

Table 5.1: Overbounded envelopes for different lengths of the sliding time windows

Therefore, the combination of the expressions (5.2) and (5.3) in fact is

$$\begin{aligned}&U\left(y\left(0\right),Y\left(0\right)\right)U\left(u\left(0\right),U\left(0\right)\right)...U\left(u\left(2w-1\right),U\left(2w-1\right)\right)U\left(p_i,P_i\right)|_{0\ldots w-1} \qquad (5.5)\\&U\left(p_i,P_i\right)|_{w\ldots 2w-1}E\left(y\left(2w\right),Y\left(2w\right)\right)\left(y\left(2w\right)=f\left(y\left(0\right),u\left(0\right),\ldots,u\left(2w-1\right),p_i\right)\right)\end{aligned}$$

which means that the parameters of the system are allowed to vary in time at a speed depending on the size of the window. The smaller is the window, the higher is the allowed variation speed and the wider are the exact envelopes.

The overbounded envelope using a particular window length includes the exact envelope corresponding to that window length, which includes the exact envelope for the time invariant system. Therefore, that envelope is also an overbounded one for the time invariant system.

In conclusion, the use of time windows to obtain an overbounded envelope for a time invariant system reduces the computation time and allows the simulation to be recursive but, on the other hand, increases the minimum achievable width for this envelope. In [74], it has been proved that the minimum window length that achieves to produce a non-diverging envelope is in the order of the time constant of the system. It becomes not practical to increase the time window length beyond this value because it has been observed that the computing effort increases faster than the precision.

5.3.2 Underbounded envelope

The problem is more complicated in the case of the underbounded envelope. The envelope at a time point w has the following semantics:

$$U\left(y\left(w\right),Y\left(w\right)\right)E\left(u\left(0\right),U\left(0\right)\right)...E\left(u\left(w-1\right),U\left(w-1\right)\right)E\left(p_i,P_i\right)|_{0...w-1} \qquad (5.6)$$
$$E\left(y\left(0\right),Y\left(0\right)\right)\left(y\left(w\right)=f\left(y\left(0\right),u\left(0\right),\ldots,u\left(w-1\right),p_i\right)\right)$$

If it is used as the starting point to compute the envelope at time point $2w$, the semantics of this envelope is

$$U\left(y\left(2w\right),Y\left(2w\right)\right)E\left(u\left(w\right),U\left(w\right)\right)...E\left(u\left(2w-1\right),U\left(2w-1\right)\right) \qquad (5.7)$$
$$E\left(p_i,P_i\right)|_{w...2w-1}E\left(y\left(w\right),Y\left(w\right)\right)\left(y\left(2w\right)=f\left(y\left(w\right),u\left(w\right),\ldots,u\left(2w-1\right),p_i\right)\right)$$

which can not be combined with (5.6) because $E\left(p_i,P_i\right)|_{0...w-1}$ and $E\left(p_i,P_i\right)|_{w...2w-1}$ do not imply $E\left(p_i,P_i\right)|_{0...2w-1}$. The envelope obtained in this case is included in the exact envelope of the corresponding time variant system, which includes the exact envelope of the time invariant system. Therefore, the relation between the first and the last of these envelopes is not clear.

A possible approach to this problem is to compute envelopes with other semantics such that when they are combined the resulting semantics is the one of the underbounded envelope. This is the case when the semantics of the envelope from 0 to $w-1$ (5.6) is combined with the semantics

$$U\left(y\left(2w\right),Y\left(2w\right)\right)U\left(p_i,P_i\right)|_{w...2w-1}E\left(u\left(w\right),U\left(w\right)\right)...E\left(u\left(2w-1\right),U\left(2w-1\right)\right) \qquad (5.8)$$

$$E\left(y\left(w\right),Y\left(w\right)\right)\left(y\left(2w\right)=f\left(y\left(w\right),u\left(w\right),\ldots,u\left(2w-1\right),p_i\right)\right)$$

of the envelope from w to $2w-1$. The resulting semantics is

$$U\left(y\left(2w\right),Y\left(2w\right)\right)E\left(p_i,P_i\right)|_{0\ldots 2w-1}E\left(u\left(0\right),U\left(0\right)\right)\ldots E\left(u\left(2w-1\right),U\left(2w-1\right)\right) \quad (5.9)$$
$$E\left(y\left(0\right),Y\left(0\right)\right)\left(y\left(2w\right)=f\left(y\left(0\right),u\left(0\right),\ldots,u\left(2w-1\right),p_i\right)\right)$$

which is the one of the underbounded envelope.

It is worth noticing that $y\left(2w\right)$ in (5.8) can not be determined using global optimisation methods. The envelopes obtained by this approach may be much more underbounded than the exact envelopes for the time invariant system because the use of a weaker semantics implies a loss of information. This is shown in tables 5.2 and 5.3.

window length	underbounded envelope
50	$[0.1584, 1.6356]$
25	$[0.6341, 1.1255]$

Table 5.2: Underbounded envelopes at time point $t = 50$ s

window length	underbounded envelope
20	$[0.6847, 1.9925]$
10	$[1.0341, 1.4736]$

Table 5.3: Underbounded envelopes at time point $t = 20$ s

5.4 Summary

This chapter presents a simulator that generates error-bounded envelopes. The generation of these envelopes needs a computation effort that depends on the desired error between them.

This effort increases at each step of the simulation. A way to stabilise the effort at each time step is by using sliding time windows. In this case, the system is considered time variant and the envelopes obtained correspond to such a system, not to the time invariant one. The allowed speed of the variation of the parameters depends on the length of the window.

The exact envelope of the time invariant system is included in the exact envelope of the time variant system, so an overbounded envelope of the time variant system is an overbounded envelope of the time invariant system. Such a comparison can not be established for the underbounded envelopes. Therefore, the underbounded envelope of the time variant system may not be an underbounded envelope of the time invariant system.

A possible solution is the use of a different semantics for the determination of the underbounded envelope. This approach has been assessed but the resulting envelopes are highly underbounded.

Chapter 6

Application of error-bounded envelopes to the fault detection problem

6.1 Introduction

The envelopes can be used for many applications: estimation of non-measurable variables, predictive control, analysis of uncertain systems, design of controllers, etc. One of them is FD. This chapter gives an introduction to this field in order to situate the envelopes in the context of the different techniques that are used in FD. A special emphasis is made on the aspects related to the applications of envelopes to it.

The properties of the envelopes have radical implications on the properties of the FD systems built upon them. If the envelopes have unknown properties, it is unknown if there will be false or missed alarms or not. If the envelopes are overbounded (resp. underbounded), there will be missed (resp. false) alarms. Finally, if they are neither complete nor sound there will be false and missed alarms. Therefore, the properties of the envelopes are very important when they are used for FD as they indicate the possibility to have, or not, missed or false alarms.

Moreover, the ratio of missed (resp. false) alarms depends on the degree of overbounding (resp. underbounding). So, in this case, it may not be sufficient to exhibit the properties of the envelopes in a binary manner and it is necessary to provide some kind of measure of the degree of overbounding or underbounding in order to measure the sensitivity of the FD system. This is definitely an important problem which opens new perspectives for approaching imprecise model-based FD.

An approach to this problem is given by the error-bounded envelopes. If the error between the envelopes is reduced to the minimum, then the ratio of missed or false alarms will be also the minimum.

However, it will be seen that for FD applications it is not necessary to work with a small error except in the case when the measure is between the two envelopes. Therefore, the desired error can be taken as a dynamic parameter of the simulation and its value can be updated on-line. This reduces significantly the computational effort needed and facilitates the use of this method in real-time systems. MIS is adapted for this particular application so it includes some new features.

6.2 The fault detection problem

In the world of today it is not enough to produce what the markets demand. It has to be produced at the lowest possible price. Hence, the reduction of the production costs is a major concern in the industry. One way to reduce costs is producing as much time as possible hence reducing to the minimum the time in which the production is stopped. These stop times occur when the process is under reparation after a breakdown or when there has to be a maintenance operation. In the ideal situation, maintenance has to be done as late as possible in order to reduce the number of maintenance operations but not too late because then there can be a breakdown. The traditional way to deal with this problem is by fixing a plan of maintenance operations based on the knowledge of the process. For instance, the oil of the cars is changed every 10000 km independently of the use of the car, although it is known that the life length of the oil is different if the car is used mostly in the city or in the highway.

Another way to deal with the maintenance problem is by observing the behaviour of the process and detecting changes in it. This is a FD problem, considering a fault as an unexpected change in a system, such as a component malfunction and variations in the operating condition, that tend to degrade overall system performance [21]. Consequences of degradation are not only economic loss, they can be extremely serious in terms of environmental impact or danger for the population. In order to design a reliable, fault tolerant system, or to maintain a high level of performance for complex systems it is crucial that such changes are detected promptly and diagnosed so that correcting action can be taken to reconfigure the system and accommodate the change.

It is important to distinguish between faults and failures. A fault denotes a malfunction and may denote something tolerable in its early stage while a failure suggests a complete breakdown, a catastrophe. However, the early detection of a fault can help to avoid major plant breakdowns and catastrophes, that is failures. Faults are specially difficult to detect in controlled systems because the task of the controller is keeping the output of the system at the set point even if there are faults [70], which in this case are viewed as perturbations.

There are several tasks related to the FDI (Fault Detection and Isolation) field [44]:

- Fault detection: the indication that something is going wrong in the system.
- Fault diagnosis:
  - Fault isolation: the determination of the kind and exact location of the fault.
  - Fault identification: the determination of the size, type, nature, magnitude, cause of the fault.
- Fault accommodation: the reconfiguration of the system using healthy components.

This work is focused on FD, although it will be seen that in the future it can be extended to fault isolation.

The performance of a FD system can be measured with the following indices:

- Missed alarms ratio. There is a missed alarm when there is a fault and the FD system does not detect it.
- False alarms ratio. There is a false alarm when the FD system indicates a fault and the system is not faulty.

- Detection delay. It is the delay between the appearance and the detection of a fault.

An ideal FD system detects faults as soon as they appear and there are neither missed alarms nor false alarms. A more realistic goal is to reduce these three indices to the minimum.

6.3 Physical redundancy: the traditional approach to fault detection

In the past, the FD problem was approached by several techniques: frequency spectrum analysis, fault dictionary approach, limit and trend checking, etc. One of them is highlighted here: physical redundancy. Some examples of physical redundancy are:

- Installation of multiple sensors. Different sensors are used to measure the same variable. If there is a serious discrepancy between the different measurements, it is clear that at least one sensor is faulty.

- Installation of multiple hardware or software. For critical operations like the launching of a spacecraft, hardware and software is doubled or tripled to increase safety.

In all these cases there is a comparison between several devices and it is assumed that their behaviour is the normal one if all of them behave the same. In other words, the behaviour of the system that is necessary for the work to be done is compared with the behaviour of another system that is used only for comparing purposes (see figure 6-1). The simplest and most efficient way to generate this reference behaviour is working with an exact copy of the real system in the same environmental conditions. This way provides the real behaviour. Nevertheless, in many cases this is not practical because of economical reasons or other kinds of constraints. An alternative to overcome these problems is the use of a model to generate the reference behaviour, i.e. to perform analytical redundancy.

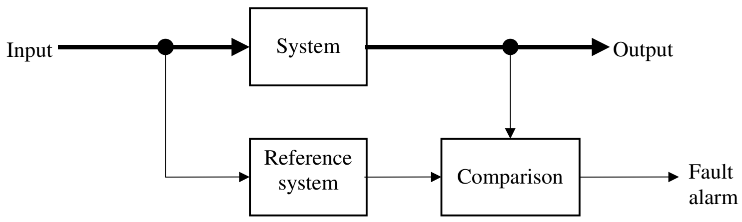


Figure 6-1: Physical redundancy

6.4 Analytical redundancy: the modern approach to fault detection

Any kind of model can be used as a reference to see if the system is faulty or not. For instance, an approach to FD can be performed by using an expert system based on the human knowledge of the system in normal and faulty situations. In the literature, the approaches of this kind are referred to as knowledge-based approaches, while the approaches based on a mathematical model of the system are called model-based approaches. All these approaches (physical redundancy, knowledge-based approaches and model-based ones) can be used for FD. Even, for complex systems, they can be combined into a single FD system.

This work is based on model-based approaches [27], which aim to detect faults by comparing the measurement of a variable and its prediction obtained by simulation using the mathematical model of the system [13, 18, 45, 57] (see figure 6-2).

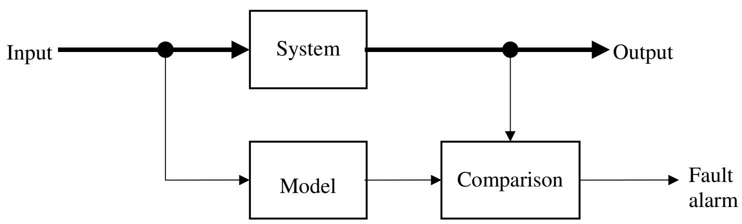


Figure 6-2: Analytical redundancy

A quantitative model and a usual simulator for this kind of models can be used to predict the values of the variables of the system. In the ideal case, the measurements and the predicted values match exactly if the behaviour of the system is normal, so the system is said to be faulty when there is a difference between them. Nevertheless, section 2.4 has shown that, in general, these values are different in the reality due to the uncertainties of the system and the sensors. Therefore, in the real case, the rule based on the exact match between the predicted value and the measured one is no more valid. One approach to this problem is by evaluating the residual, which in this case would be the difference difference between these two values, using statistical or interval-based methods.

6.4.1 Passive generation of residuals

This approach indicates a fault when the residual is greater than a threshold (see figure 6-3). The main problem now is choosing the threshold. If it is too small there will be false alarms and if it is too large there will be missed alarms, as can be seen in the example of figure 6-4.

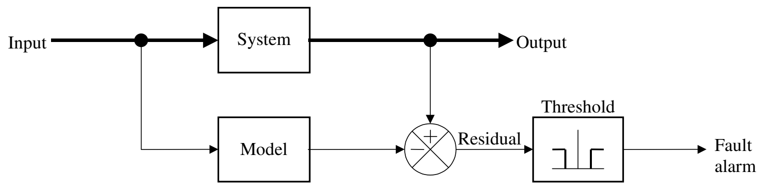


Figure 6-3: Analytical redundancy using a fixed threshold.

The threshold can be computed trying to take into account the uncertainties and imprecisions that existed in the modelling process, the noise of the sensors, etc. This fixed threshold usually is not very useful for dynamic systems because the model is an approximation that is better or worse depending on the operating point of the system. For instance, if the system is non-linear and it is approximated by a linear model, the linearisation is done around an operating point and hence the model is a good approximation only for points close to this nominal operating point. In this case, it is better to compute a new threshold every time step. This is an adaptive threshold (see figure 6-5).

This approach predicts the behaviour of a nominal model of the system and then the un-

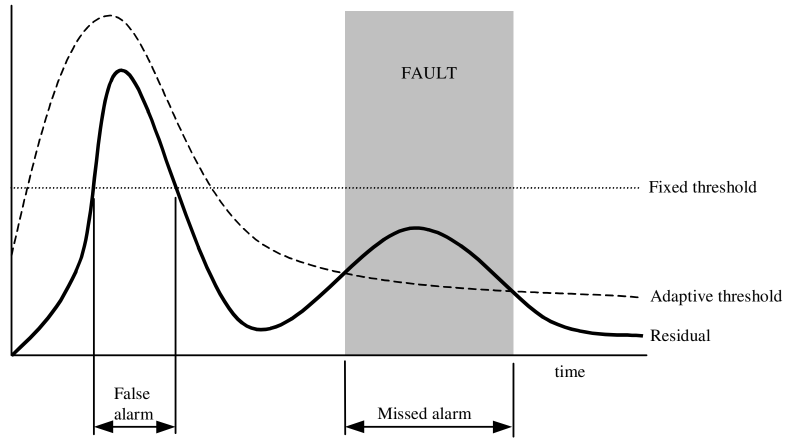


Figure 6-4: Comparison of a fixed threshold and an adaptive one.

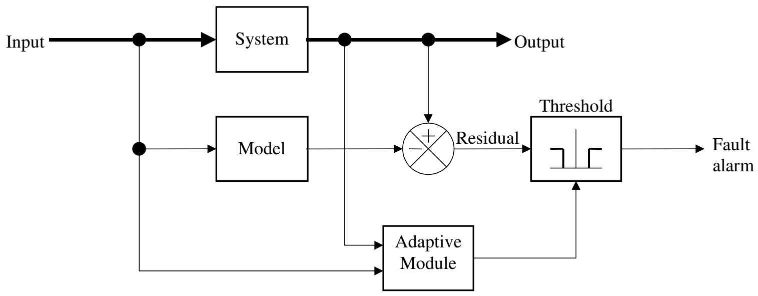


Figure 6-5: Analytical redundancy using an adaptive threshold.

certainties are managed somehow to predict the worst discrepancies between the real system and the nominal model (threshold) [49]. Therefore, the set of allowed values of the measured output a a time point consists of all the values such that their difference with respect to the predicted one is not greater than the threshold:

$$Y_N = \{y \,||y - \widehat{y}| \leq \Delta\widehat{y}\} \tag{6.1}$$

where:

- Y_N is the set of allowed values of y.
- $\widehat{y}$ is the predicted value of y.
- $\Delta\widehat{y}$ is the threshold for $\widehat{y}$.

In general, for dynamic systems, the performances of adaptive thresholds are better than the ones of fixed thresholds (see figure 6-4) but they are also more difficult to compute [35].

Envelopes

The envelopes are a way to represent the thresholds. The envelope can be obtained using a specific adaptive threshold as a tolerance that is added and subtracted to the trajectory of some nominal model:

$$Y_N = \{y \,|\widehat{y} - \Delta\widehat{y} \leq y \leq \widehat{y} + \Delta\widehat{y}\} \tag{6.2}$$

When the envelopes are used to detect faults, a fault is indicated if the measure is outside of the envelope (see figure 6-6). It could seem that using the exact envelope there are not neither missed alarms nor false alarms. This is not completely true due to the dynamics of the systems, as the measure can remain inside the envelope for some time after a fault has occurred. Therefore, the fault is not detected immediately and some time, dependent on the distance between the actual values of the system parameters and their nominal values, is needed to detect the fault. If this distance is small, the time necessary to detect the fault is larger.

The properties of an envelope are very important when they are used for FD. They determine the possibility or not to have missed or false alarms. On one hand, if an overbounded

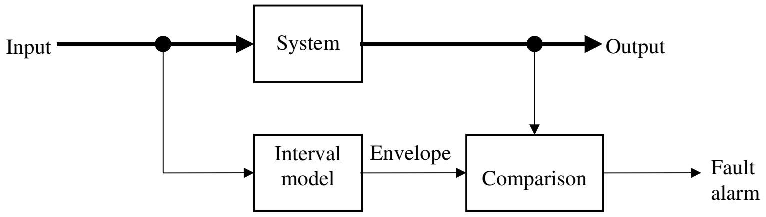


Figure 6-6: FD using envelopes.

envelope is used, there can be missed alarms but not false alarms. On the other hand, when an underbounded envelope is used for FD, there can be false alarms but there will never be missed alarms. In each particular case it has to be assessed if it is worse to have either false alarms or missed alarms, but in any case the goal is to produce either a minimum overbounded envelope or a minimum underbounded envelope.

6.4.2 Active generation of residuals

An alternative way to deal with this problem is the active robust generation of residuals [21]. The passive solution (fixed and adaptive thresholds) focuses on the decision making stage, i.e. the comparison of the residual and the threshold. Therefore, the threshold is critical. The active solution proposes to focus on the residual generation stage, that is to generate residuals that are very different when there is a fault and when there is not. This can be achieved computing the residuals not only by making the difference, but computing them by more complex functions that try to be insensitive to modelling uncertainties and sensitive to faults (see figure 6-7). In this case the threshold is not so critical.

6.5 Fault detection using error-bounded envelopes

The computation of an overbounded envelope and an underbounded one simultaneously has an interesting application in FD. These two envelopes define three zones (see figure 6-8). It is guaranteed that there is a fault when the measure is in the outer zone, outside the overbounded envelope, but the measure can be inside this envelope even when the system is faulty.

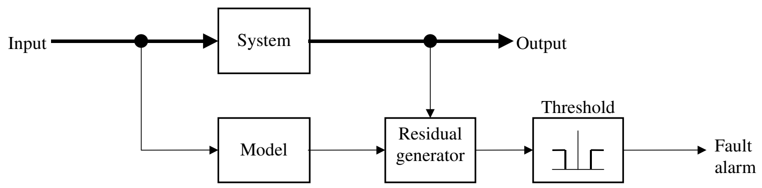


Figure 6-7: Active generation of residuals.

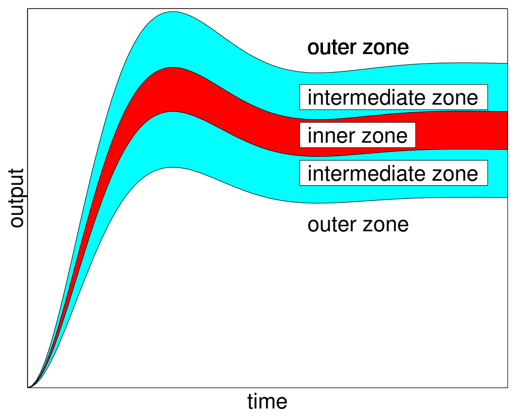


Figure 6-8: The three zones defined by error-bounded envelopes

One reason is that the overbounded envelope includes spurious points that cannot be reached by any of the systems represented by the interval model. The second reason is that the measure can remain inside the envelope for some time after a fault has occurred due to the dynamics of the system. Therefore the fault is not detected immediately and some time, dependent on the distance between the actual system parameter values and their nominal values, is needed to detect the fault. The smaller the distance, the larger the time.

Therefore, the output of a faulty system can be in any of these three zones whereas the output of a not faulty system can be in the inner zone or in the intermediate one, but never in the outer zone. This is summarised in table 6.1.

		System	
		Not faulty	Faulty
Zone	Inner	Allowed	Allowed
	Intermediate	Allowed	Allowed
	Outer	FORBIDDEN	Allowed

Table 6.1: The three zones determined by the error-bounded envelopes.

If a measurement of the output of the system is taken and it is in the outer zone, an alarm can be indicated, independently of the error between the two envelopes. If it is in the inner zone, nothing can be said about the behaviour of the system. It is not known if it is faulty or not. This is valid whatever is the distance between the two envelopes. Finally, if the measurement is in the intermediate zone, possibly a fault should be detected if closer envelopes are calculated and actually the measurement is in the outer zone.

This comparison is made by MIS adapted to the detection of faults. It generates error-bounded envelopes and compare them to the measurements in order to detect faults. The results of FD are indicated in the following way:

- $fault = 1$. The measurement is in the outer zone. The fault is detected.
- $fault = 0.5$. The measurement is in the intermediate zone. The fault is not detected.
- $fault = 0$. The measurement is in the inner zone. The fault is not detected.

6.5.1 Example

MIS is used for the example at section 5.2.3 to detect that the generic first order system

$$y_n = \left(1 - \frac{T}{\tau(t)}\right) y_{n-1} + \frac{k(t)\,T}{\tau(t)} u_{n-1} \tag{6.3}$$

is faulty when its parameters degrade with time:

- Static gain:

  $$k(t) = k_0 + m_k t \tag{6.4}$$

  with $k_0 = 1.05$ and $m_k = -0.001\ \frac{1}{\mathrm{s}}$.

- Time constant:

  $$\tau(t) = \tau_0 + m_\tau t \tag{6.5}$$

  with $\tau_0 = 5$ s and $m_\tau = -0.04\ \frac{\mathrm{s}}{\mathrm{s}}$.

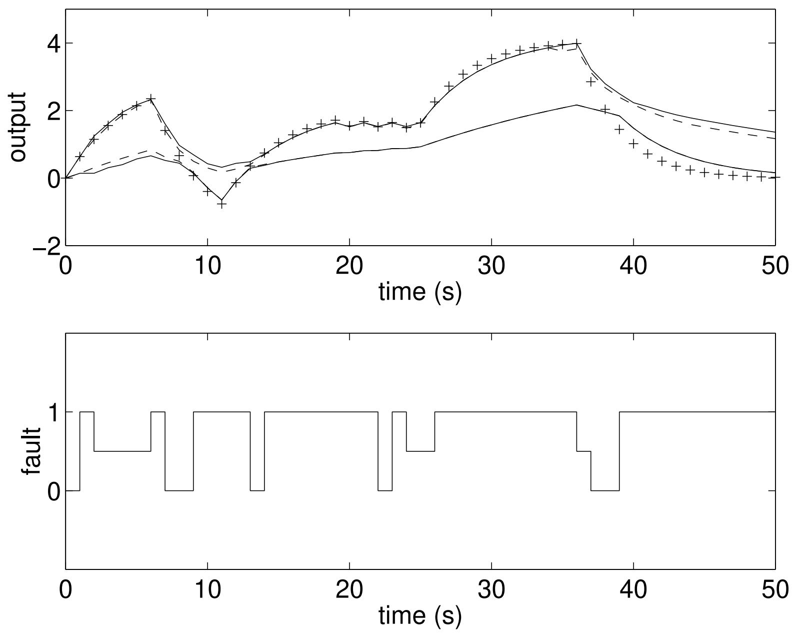


Figure 6-9: FD with error-bounded envelopes.

As the nominal values are $k = [0.95, 1.05]$ and $\tau = [5, 20]$ s, initially the system is not faulty and the time constant gets more and more far from its nominal value with time. This is shown in figure 6-10.

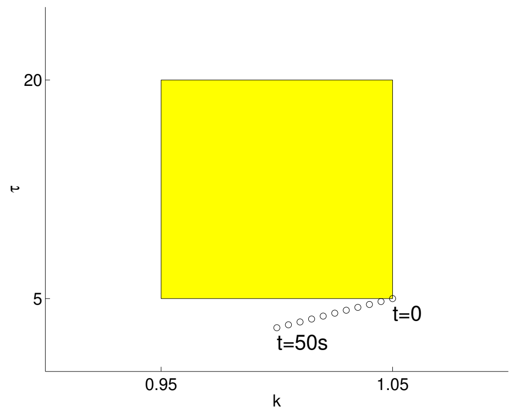


Figure 6-10: Faulty model represented in its parameter space.

Figure 6-9 shows the results. The upper half of the figure shows the error-bounded envelopes (the overbounded envelope in solid line and the underbounded envelope in dashed line) jointly with the output of the faulty system (crosses). The lower half of the figure shows the FD results.

6.5.2 Conclusions

The example above shows that in the cases where the measurement is in the outer or in the inner zone it would probably be in the same zone if less calculations were performed and the distance between the envelopes was larger. And, on the other hand, more computations should be performed to get the envelopes closer in the case where the measurement is in the intermediate zone. This is what is introduced in MIS in the following section.

6.6 Fault detection using dynamic error-bounded envelopes

To reduce the amount of measurements in the intermediate zone is necessary to calculate error-bounded envelopes with a small error. This requires a high computation effort that is unnecessary in many cases. This effort only has to be done when the measurement is in the intermediate zone and until the measurement belongs to the inner or to the outer zones.

Therefore, when a fixed error between the two envelopes is used, more calculations than needed are performed in some cases while less calculations than needed are performed in other cases. A new feature that is introduced in MIS avoids this problem. Now a measurement of the output of the system is needed. This measurement is used to stop the iterations instead of using a maximum distance between the envelopes. The iterations stop when the measurement is at the outer zone or at the inner zone, i.,e. it is not in the intermediate zone. This is done at the new function **stop_condition**, which substitutes the one of the same name in the branch and bound algorithm of section 5.2.1:

---

function $b =$**stop_condition**$(external, internal, measurement)$

IF $measurement \geq \overline{external}$ THEN

$b = true$

ELSEIF $measurement \leq \overline{internal}$ AND $measurement \geq \underline{internal}$ THEN

$b = true$

ELSEIF $measurement \leq \underline{external}$ THEN

$b = true$

ELSE

$b = false$

END

This new algorithm also has been implemented using Matlab, Maple and C++.

6.6.1 Example

MIS is applied to the example of section 6.5.1. The results of the simulation are shown in figure 6-11. Comparing it with figure 6-9, it may be observed that FD results have been enhanced: measurements in the intermediate zone have disappeared. In addition, these results have been obtained with a smaller computational effort. This can be seen observing that there are several time points where the overbounded envelope is much more overbounded than in figure 6-9. In fact, the overbounded envelope is outside the figure at some of these points because the scale of figure 6-9 has been mantained in figure 6-11 to facilitate comparisons. At these points the overbounded envelope is so overbounded because at one of the first iterations of the algorithm already has been seen that the measurement is not in the intermediate zone so it is not necessary to obtain better (less overbounded) approximations of it.

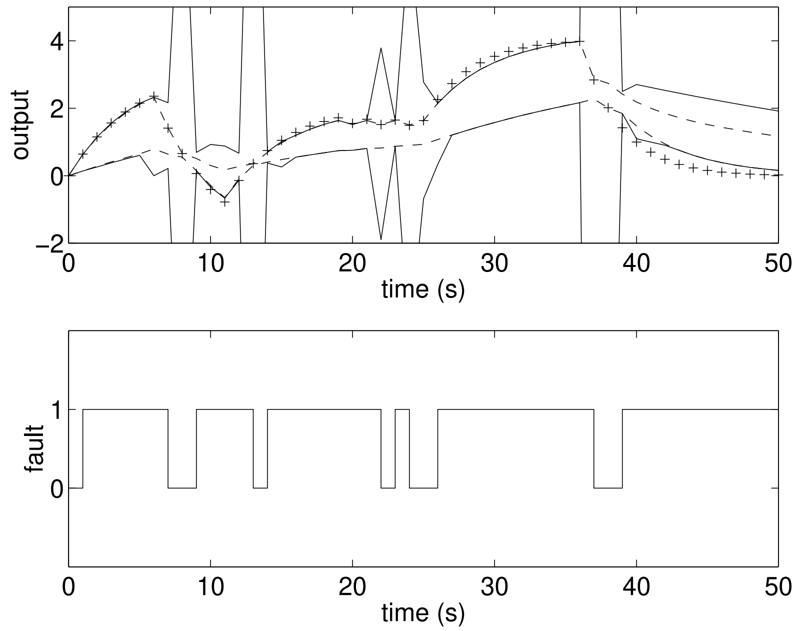


Figure 6-11: FD using error-bounded envelopes with variable error.

6.6.2 Conclusions

The measurements can be used to stop the branch and bound algorithm that calculates the envelopes. This saves computational efforts when the measurement is in the inner or in the outer zones and avoids measurements in the intermediate zone. This has been implemented in MIS.

MIS computes the envelopes with respect to the initial state and hence the computing time needed increases at every step of the simulation. Next section shows that this time can be stabilised if sliding time windows are used.

6.7 Use of sliding time windows

In section 5.3, the problems related to the use of sliding time windows in the simulation have been discussed. These problems are due to the use of error-bounded envelopes at the beginning of the window to calculate error-bounded envelopes at the end of the window. They are intrinsic with the simulation: the output is obtained starting from the model of the system, its initial state and the input, i.e. the measurement of the output is used only for the initial state.

When the error-bounded envelopes are used for FD, the measurements are needed during the simulation if they are used to stop the algorithm. Therefore, there is another option in this case. The measurements can be used at the beginning of a window to calculate the error-bounded envelopes at the end of that window. There are no semantical differences between the calculation of the envelopes starting from the measured initial state and the calculation of the envelopes at the end of a window starting from the measured state of the system at the beginning of that window. Therefore, the use of the measurements reduces the computation effort and, simultaneously, avoids the semantic problems that appear when the measurements are not used.

Some new features have been added to MIS. Now there is the possibility to use sliding time windows and there are several options for the starting point of each window:

- The error-bounded envelopes.
- The measurement.
- The error-bounded envelopes or the measurement depending on the system being faulty or not, respectively. This possibility is the one used by Ca$\sim$En and avoids that the envelopes "follow" the measurements when the system is known to be faulty.

The first option is not very adequate when the condition to stop the algorithm is only the location of the measurement with respect to the envelopes because the error between the envelopes at the beginning of the window may be very large and the consequence is a large error at the end of the window. This can be solved using a double condition to stop the algorithm: the measurement is not in the intermediate zone and the error between the envelopes is lower than a fixed one.

The use of sliding time windows allow to use these simulators for long simulations because the computational effort no more increases with time. The only problem is to choose the length of the window because the computing time needed at every step and the allowed speed for the variation of the parameters depend on it. In [74] there is a discussion about the window length and the distance between the envelopes that are obtained using sliding time windows and the envelopes of the time invariant system. The conclusion is that a length in the order of the time constant of the system is recommended to avoid that this distance increases too fast.

It is worth noticing that, on one hand, the measurement at a time point is a real number and, on the other hand, it has an imprecision associated with it, so it should be better expressed with an interval. This should to be taken into account when this measurement is compared with the error-bounded envelopes to detect faults and when it is used as the starting point of a window. In the former case, a comparison between the envelopes and an interval has to be performed. This is a task that will be considered for the future work. In the latter case, the starting point of the window has to be an interval. For this reason, MIS uses interval-valued measurements to generate the envelopes, although the measurements are still considered as real numbers when they are compared with the envelopes in order to detect faults.

6.8 Comparison of different sliding time windows

The application of sliding time windows shows that good results can be obtained using adequate window lengths. The problem is choosing the adequate window length. It depends on several factors: the fault that is to be detected, the noise in the measurements, the use of interval measurements or not, etc. To help the user to choose the window length, a new feature has been added to MIS: the use of multiple windows. It allows the user to simulate not only with a single window length but also with several window lengths at the same time. In this case, the result of the FD may be different for the different window lengths. There is clearly a fault if it is indicated by all of them. If there are not indications of fault, possibly there is a fault but there are no means to detect it, at least with these tools.

6.8.1 Example

MIS is used to detect faults in the example of section 6.5.1. Now the simulator uses sliding time windows to generate error-bounded envelopes. The measurements are obtained from the faulty system

$$y_n = \left(1 - \frac{T}{\tau(t)}\right) y_{n-1} + \frac{k(t)\,T}{\tau(t)} u_{n-1} \tag{6.6}$$

with:

- static gain:
$$k(t) = k_0 + m_k t \tag{6.7}$$
where
  - $k_0 = 1.05$
  - $m_k = -0.001\ \frac{1}{\mathrm{s}}$.
- time constant:
$$\tau(t) = \tau_0 + m_\tau t \tag{6.8}$$
where
  - $\tau_0 = 5.4$ s
  - $m_\tau = -0.04\ \frac{\mathrm{s}}{\mathrm{s}}$.
- initial state: $y_0 = 0$
- sampling time: $T = 1$ s

which is a system that is degrading with time. It belongs to the nominal system until the time instant $t = 10$ s and after this time point it becomes faulty, as its is shown in figure 6-12.

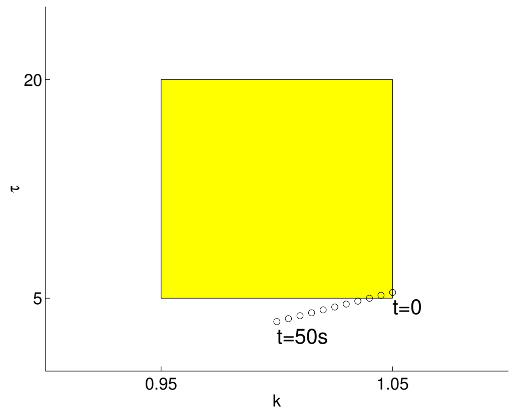


Figure 6-12: System that is faulty after $t = 10$ s.

When the envelopes are computed starting from the initial state it can be considered that sliding time windows are not used or that they are used and the windows are longer than the simulation. That is the case of sliding time windows of length $w = \infty$. The results of the FD based on the simulation of this model and $w = \infty$, are shown in figure 6-13 with the label $w = 50$ s.

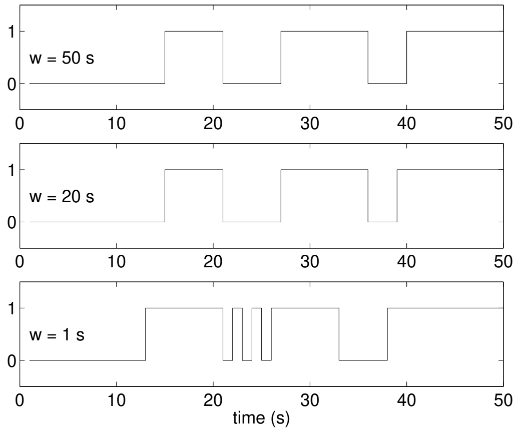


Figure 6-13: FD using window lengths 1, 20 s and ∞.

As the time constant is $\tau = [5, 20]$ s and a window length in the order of the time constant is recommended in [74], the results using a window length $w = 20$ s are also shown in figure 6-13. In this case, the error-bounded envelopes at the end of the window are computed starting from the measurements at the beginning of the window. It can be observed that the results are similar when an adequate shorter window length is used.

However, if the window is too short, the results are different. This is shown in the same figure when a window length $w = 1$ s is used. In this case, the number of transitions between 1 (the fault is detected) and 0 (the fault is not detected) is larger. This happens because, when the sliding time window is too short, the dependency of the envelopes with respect to the last measurement is very high and hence they "follow" the measurements.

This figure shows that there are not false alarms but, on the other hand, faults can not be detected in some periods of time independently of the window length used.

As there can be some uncertainty in the measurements, new simulations taking it into account are performed. Assume that this uncertainty can be represented by an additive random number in $[-0.1, 0.1]$, then the noisy measurement at a time point t ($y_{mn}(t)$) can be obtained from the measurement without noise at a time point $(y_m(t))$:

$$y_{mn}(t) = y_m(t) + \mathrm{rand}([-0.1, 0.1]) = y_m(t) - 0.1 + 0.2 \cdot \mathrm{rand}([0,1]) \tag{6.9}$$

being $\mathrm{rand}([x,y])$ a random number between x and y. Therefore, a noisy measurement vector y_{mn} is used for FD and to generate the envelopes. In figure 6-14 there are represented the noisy measurement vector and the noise that has been used to obtain it.

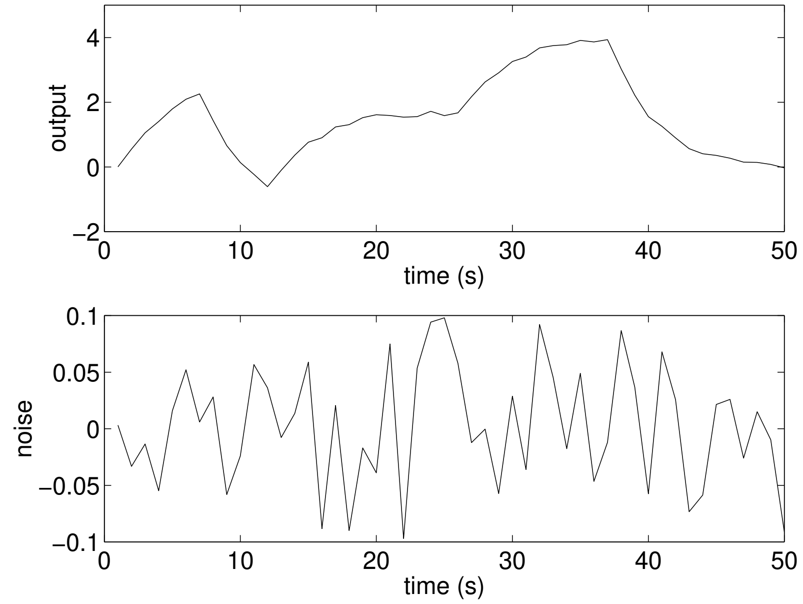


Figure 6-14: Measurement with additive noise.

The results of FD for window lengths 1, 10, 20 s and ∞ are shown in figure 6-15. The results using window lengths 20 s and ∞ are similar: both detect faults after $t = 10$ s and do not detect faults before this time point, as there is not any fault yet. It happens the same with window length $w = 10$ s, but $w = 1$ s detects faults even when there are not, i.e. it generates false alarms.

Finally, the uncertainty in the measurements is considered when the envelopes are generated and an interval measurement vector $y_{im} = [y_{mn} - 0.1, y_{mn} + 0.1]$ is used at the beginning of the windows. The results of this FD for $w = \infty$ and $w = 20$ s again are similar. They are represented in figure 6-16. On the other hand, for $w = 1$ s the results are worse, as faults are not detected most of the time.

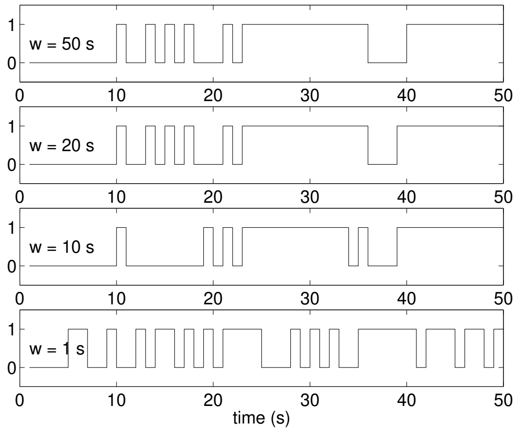


Figure 6-15: FD using noisy measurements and window lengths 1, 10, 20 s and ∞.

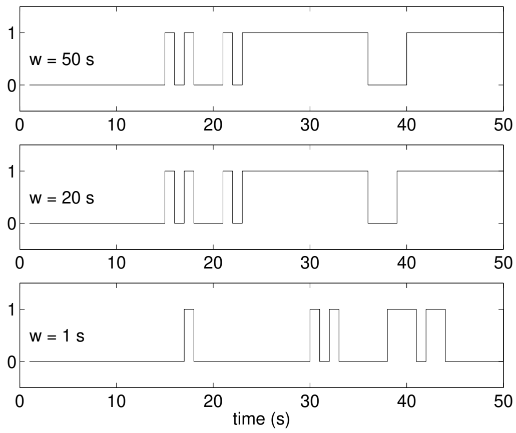


Figure 6-16: FD using interval measurements and window lengths 1, 20 s and ∞.

6.8.2 Conclusions

The exact envelope for a time invariant system is obtained simulating from a starting point, that is with an infinity window length. This needs a high computation effort which can be avoided using sliding time windows. In this case it is allowed that the parameters of the system vary at a certain speed depending on the length of the window, i.e. the system is considered time variant.

The error-bounded envelopes at the end of a window can be computed starting from:

- the error-bounded envelopes at the beginning of that window.
- the measurements at the beginning of that window, which are represented by means of intervals if there is some uncertainty in them.
- a combination of both previous options.

The first option is the only one for simulation, as there are not available measurements. In the case of FD, the other options are valid alternatives as the measurements are needed. They are compared to the envelopes to decide if either there is a fault or nothing about faults can be said.

6.9 Summary

One approach to the problem of the detection of faults is the redundancy: a system is compared with a reference one and a fault is detected when there are discrepancies. This reference can be another system (physical redundancy) or a model of the system (analytical redundancy). In the latter case, the reference behaviour is generated through simulation.

As a quantitative model is an approximate representation of the system, the real system and the model have different behaviours and there are always discrepancies between them. This is a source of false alarms. One way to approach this problem is based on the generation of robust residuals. Another one is using thresholds (fixed or adaptive) to decide between normal and abnormal discrepancies.

An envelope can be seen as a combination of the simulated output of a nominal model and a threshold. In this case, a fault is indicated if the measure is outside of the envelope.

The exact envelope for a time invariant system is obtained simulating from a starting point. When sliding time windows are used the computation effort is lower, but the system is considered time variant.

The are several options for the starting point at the beginning of a window when sliding time windows are used to compute error-bounded envelopes.

There are also several options for the condition to stop the algorithm:

- the error between the envelopes is lower than a fixed one.
- the measurement is not in the intermediate zone.
- a combination of both previous options.

So, there are several different possibilities when sliding time windows are used. Results are not good for all the possible combinations. For instance, if the error-bounded envelopes at the beginning of the window are used and the condition to stop is that the measurement is not in the intermediate zone, the distance between the envelopes at the beginning of the window may be very large. Then, the distance between the envelopes at the end of the window can not be small and possibly the measurement will be in the intermediate zone.

The results of the FD based on envelopes and sliding time windows show that adequate lengths of the sliding time windows allow to obtain useful results with a much lower computational effort than the one needed for infinity lengths.

The capability to detect a particular fault depends on several factors. One of them is the length of the window. If it is too short, the output of the simulation highly depends on the last measurement. If it is noisy, which is a common situation, the FD system can produce false alarms and missed alarms in this case. If it is too long, the computing effort is too high. On the other hand, a degradation of the parameters of the system can only be detected if the window is sufficiently long.

To help the user to deal with the window length, MIS includes the possibility to use several window lengths simultaneously. This filters in some way the results provided by each of the different window lengths.

Chapter 7

Practical cases

7.1 Introduction

This chapter presents some examples that have been used to test and validate MIS. These tasks are performed by applying it to FD.

The first example, the TIGER 2nd stage nozzles system, comes from the Esprit project TIGER (see section 1.1). It is a real example with real data.

The second example is based on the two tanks benchmark [60]. It is used in the framework of an exchange between LEA-SICA (Laboratori Europeu Associat - Sistemes Intel·ligents i Control Avançat) and the Technische Universität Hamburg-Harburg within the PROCOPE programme as a benchmark to allow the comparison between different approaches to FD and fault diagnosis.

In the following, there is a description of these examples and some illustrative FD results.

7.2 TIGER 2nd stage nozzles system

7.2.1 Introduction

This example comes from the TIGER project [62]. The goal of that project was to detect and diagnose faults in gas turbines. These devices are very complex and include many subsystems: compressor, fuel system, cooling air, cooling water, hydraulic system, inlet guide vanes, lubrication oil system, steam injection system, etc. To deal with this task, data of about 600 analog or digital variables are collected once per second.

7.2.2 Model

One of the subsystems of two shafts gas turbines is the 2nd stage nozzles system. It is formed by the nozzles, which are used to open and close the air flow through the turbine, and their associated actuator and controller. This is represented in figure 7-1, where the names of the variables are the ones used in the TIGER project:

- $TSNZ$ is the actual position of the nozzles.
- $TSRNZ$ is the reference position of the nozzles.
- $TANZ$ is the output of the controller and the input to the actuator of the nozzles.

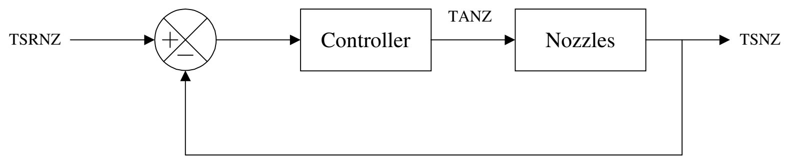


Figure 7-1: The nozzles subsystem.

The controller is modelled by the following difference equation:

$$TANZ_n = na_1 \cdot TANZ_{n-1} + na_2\,(TSRNZ_n - TSNZ_n) + na_3\,(TSRNZ_{n-1} - TSNZ_{n-1}) + na_4 \tag{7.1}$$

where:

- the subindex n indicates the time point.
- na_i are the parameters of the model. Their values are real numbers and are not given here for confidentiality reasons.

On the other side, the set formed by the nozzles and their actuator is modelled by means of the following difference equation:

$$TSNZ_n = TSNZ_{n-1} + ns_1 TANZ_{n-1} + ns_2 \tag{7.2}$$

where ns_i are the parameters of the model. The values of these parameters are intervals and are not given here for confidentiality reasons.

It is shown that the model of the nozzles is an interval one but the model of the controller is not. This is because there are uncertainties in the model of the nozzles that have been included in the model by means of interval parameters. However, the model of the controller is completely known.

7.2.3 Data

The values of the variables are collected once per second, so there are many data from the gas turbine. Most of these data belong to situations of normal behaviour and hence are not very interesting. But sometimes there are abnormal situations, unusual events or incidents that are more interesting. Data from these situations are saved and called "scenarios". Among them, there are scenarios with vibration problems, oil leaks, sensor failures, sensors bad positioned or calibrated, poorly tuned controllers, problems with valves, problems of gas supply, problems with the steam system, etc. The examples shown in this section use data from the following scenarios:

- Scenario 10. Normal behaviour. There is an oscillation in the reference that is perfectly followed by the nozzles.
- Scenario 3. Faulty behaviour. The nozzles are not responding. They are given a reference to close but they keep open.
- Scenario 40. Faulty behaviour. The actuator of the nozzles is not enough powerful so it can not close the airflow.

7.2.4 Related work

In the TIGER project, Ca$\sim$En [81] was used to detect and diagnose faults. The FD was performed by one of the components of Ca$\sim$En, based on a simulator that generates envelopes. This simulator is based on the method of range computation. It computes this range at each step of the simulation. If the measurement is inside the predicted envelope it is considered as a valid measurement and is used as the starting point of the envelope at the next step. If the measurement is outside the predicted envelope a fault is indicated and the envelope at the next time step is computed starting from the envelope at the current step.

One drawback of this method is that the envelope is highly dependent on the measurement. The measurements of the TIGER gas turbine are noisy, so very often the measurement is outside the envelope even when there are not faults. To avoid this, the indication of fault is filtered and a fault is indicated only if the measurement is outside the envelopes during several consecutive time points. This number of times is adjusted by the user.

Another drawback of this method is that it does not take into account the multi-incidences of the parameters in the functions of the different steps. This happens when the measurement is outside the envelope and the new envelope is computed starting from the current one. Therefore, Ca$\sim$En considers that the systems are time variant and that their variation speed can be very high. The result of it is that the envelopes are very wide after some steps, hence the measurement might come back into the envelope, even if the system is faulty. Another consequence is that Ca$\sim$En may be unable to detect that a system whose parameters are slowly degrading is faulty.

7.2.5 Fault detection results

This section shows some results obtained by MIS using real data from the TIGER gas turbine. MIS has different possibilities for the starting point of the windows and the condition to stop the algorithm. The ones that have been chosen are:

- The starting point of a window is the measurement at that time point.
- The condition to stop the algorithm is that the measurement is not in the intermediate zone between the error-bounded envelopes.

Two models have been used:

- The open loop model. Only the model of the nozzles subsystem is used. The input is $TANZ$ and the output is $TSNZ$.

- The closed loop model. The model of the nozzles and the model of the controller are used. The input is $TSRNZ$ and the output is $TSNZ$.

The results obtained by Ca$\sim$En are also shown to allow the comparison between them and the ones obtained by MIS.

Scenario 10

This scenario shows that in response to a step change in some variable, the nozzles must open. Due to the tuning of the controllers, the nozzles are given a reference ($TSRNZ$) which consists of a change with a damped oscillation. The nozzles ($TSNZ$) follow the reference so there is not any fault. This can be seen in figure 7-2, where $TSRNZ$ is the dashed curve and $TSNZ$ is the solid one. There is some offset between them.

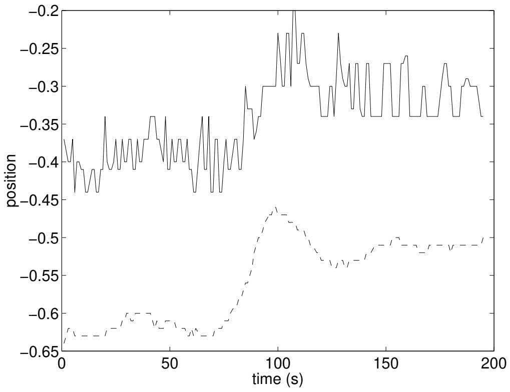


Figure 7-2: Scenario 10: $TSRNZ$ and $TSNZ$.

When this scenario is simulated with Ca$\sim$En, the obtained envelopes for $TSNZ$ are the ones at the top of figure 7-3 represented in dotted lines. The measured $TSNZ$ is also represented in solid line. If there is an indication of fault every time that the measurement is outside the envelope, in this scenario faults are indicated at some time points because the measurement of $TSNZ$ is noisy. These indications of fault are also represented at the center of figure 7-3.

To avoid these indications of fault when measurements are noisy, a filter was added to Ca$\sim$En. It consists of indicating a fault only if several consecutive measurements are outside the envelope. The amount of measurements outside the envelope to indicate a fault is adjusted empirically. In this case it was adjusted to 5. Using this filter, the indications of fault are the ones at the bottom of figure 7-3, i.e. there are not indications of fault.

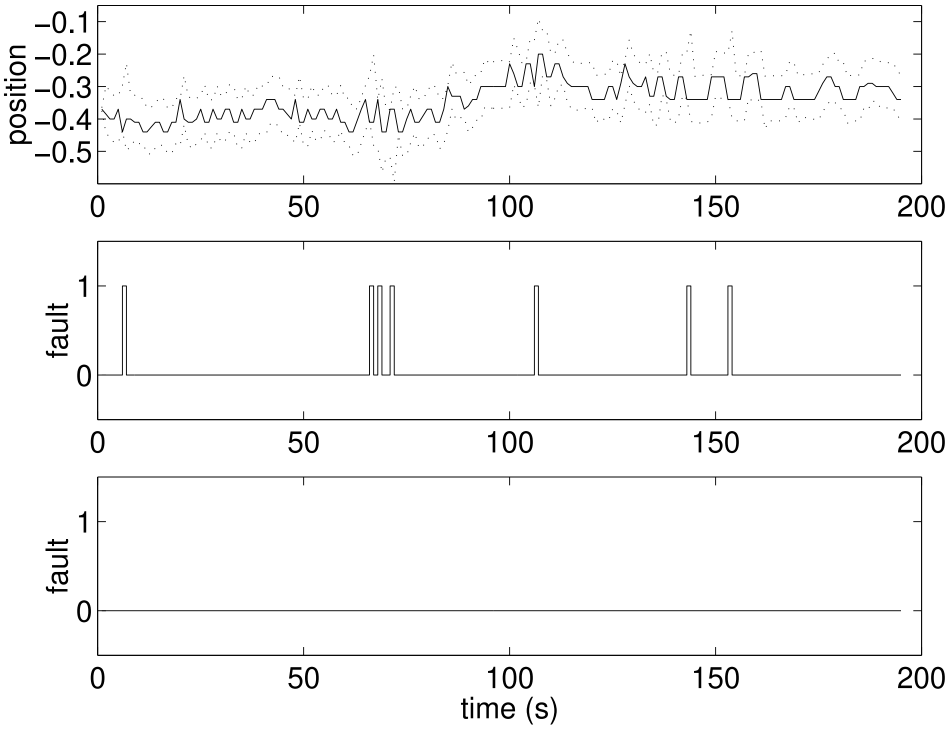


Figure 7-3: Scenario 10: (top) envelopes for $TSNZ$ and corresponding faults generated by Ca$\sim$En (center) without and (bottom) with filter.

The same scenario is assessed using MIS. First, the open loop model is used. The window lengths used are: 1, 2, 5, 10, 20 s and ∞. Figure 7-4a shows that the results of $w = 1$ s are similar to the ones of Ca$\sim$En without the filter. If the closed loop model is used, the results are the ones of figure 7-4b.

In both cases the unique window length that detects faults in this scenario that is not faulty is $w = 1$ s, hence it is considered that it is not adequate in this case.

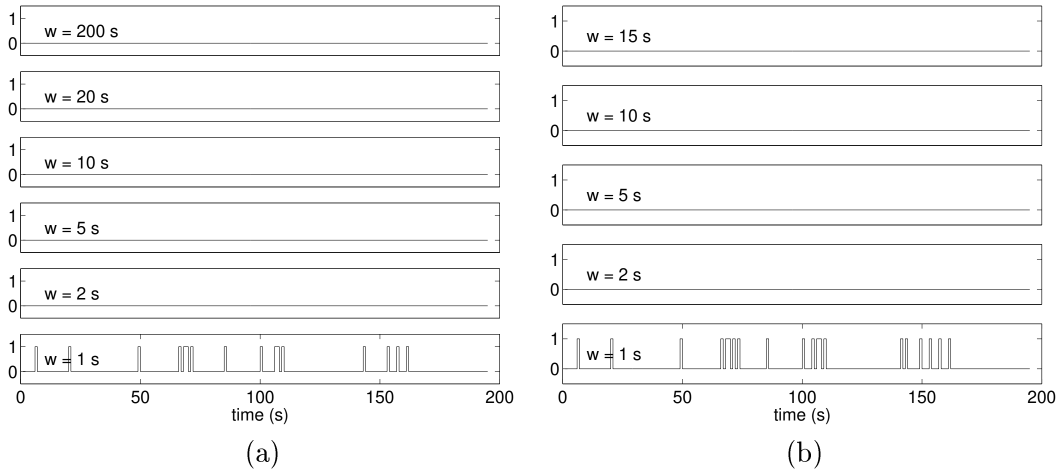


Figure 7-4: Scenario 10: indications of fault using (a) the open loop model and windows of length 1, 2, 5, 10, 20 s and ∞ and (b) the closed loop model and windows of length 1, 2, 5, 10 and 15 s.

Scenario 3

In this scenario, the reference for the nozzles $TSRNZ$ is decreasing after some time, but the position of the nozzles is steady. This can be seen in figure 7-5.

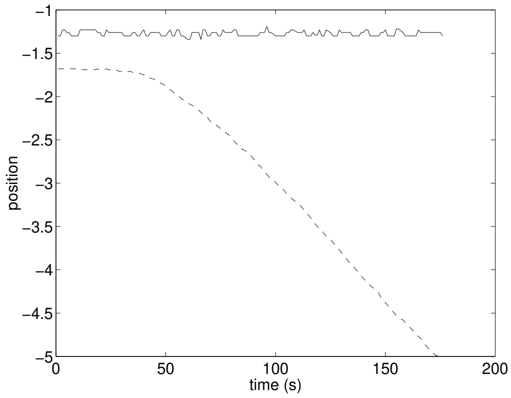


Figure 7-5: Scenario 3: $TSRNZ$ and $TSNZ$.

Figure 7-6 shows that Ca$\sim$En clearly detects the fault.

The results obtained using the open loop model and MIS with the window lengths 2, 5, 10, 20 s and ∞ are shown in figure 7-7a. Now $w = 1$ s is not used because it has been discarded in the previous scenario. Figure 7-7b shows the results using the closed loop model and the

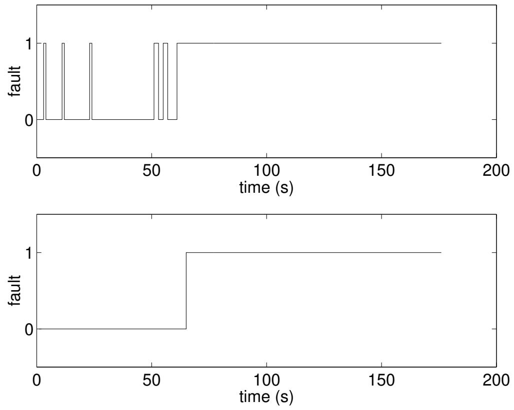
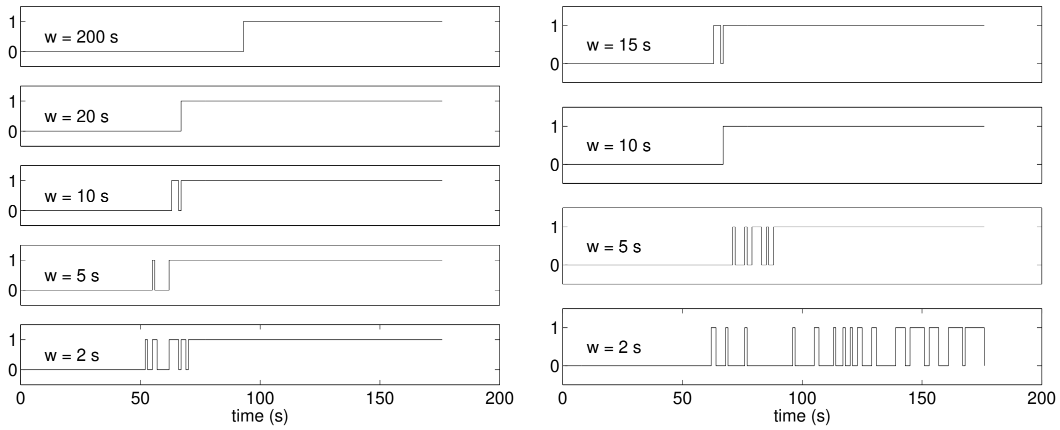


Figure 7-6: Scenario 3: faults indicated by Ca$\sim$En (top) without and (bottom) with filter.

following window lengths: 2, 5, 10 and 15 s. It shows that $w = 2$ s is not very adequate.



(a) (b)

Figure 7-7: Scenario 3: indications of fault using (a) the open loop model and windows of length 2, 5, 10, 20 s and ∞ and (b) the closed loop model and windows of length 2, 5, 10 and 15 s.

Scenario 40

This scenario had a special interest in the TIGER project, as it helped to discover a problem that was hidden before. Due to some circumstances, the nozzles are given a reference to close for a while and open again. They close, but less than expected, as it can be seen in figure 7-8. It was due to a lack of power of the actuator, as it was discovered after.

The indications of fault of Ca$\sim$En are shown in figure 7-9.

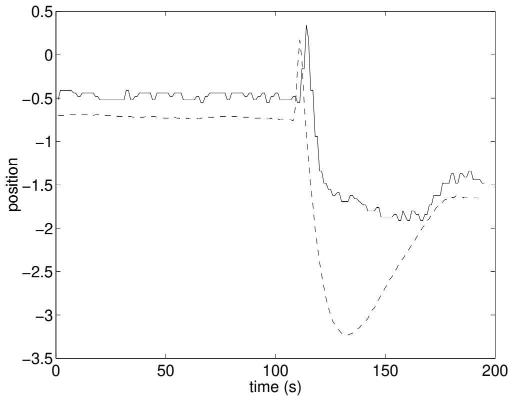


Figure 7-8: Scenario 40: $TSRNZ$ and $TSNZ$.

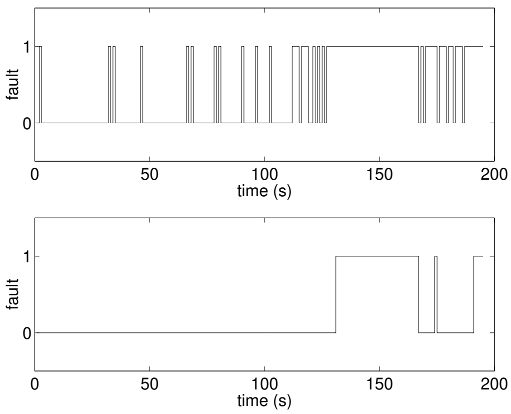


Figure 7-9: Scenario 40: faults indicated by Ca∼En (top) without and (bottom) with filter.

The results of MIS using the open loop model and windows of length 5, 10, 20, 50, 100 s and ∞ are shown in figure 7-10.

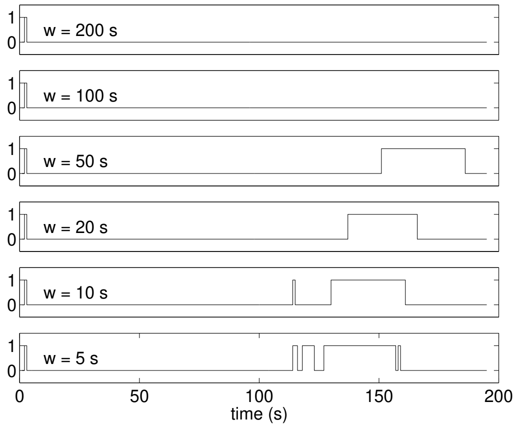


Figure 7-10: Scenario 40: indications of fault using the open loop model and windows of length 5, 10, 20, 50, 100 s and ∞.

It can be seen in this figure that the window lengths 100 s and ∞ never indicate faults.

Figure 7-11a shows the error-bounded envelopes for $w = 50$ s. The measured value of $TSNZ$ is outside the overbounded envelope during some time. This means that none of the models belonging to the interval one can give an output inside the exact envelope given the measurement that existed 50 s before and the inputs that existed during that period of time. Nevertheless, figure 7-11b shows that it is possible to have the output inside the exact envelope when 100 s are considered.

The results obtained with the open loop model are confirmed when the closed loop one is used. This is shown in figure 7-12, where windows of lengths 5, 10 and 15 s are used.

7.2.6 Conclusions

The results obtained by MIS using a window length of 1 s are similar to the ones obtained by Ca$\sim$En without the filter. These results are not good because they are highly dependent on the last measurements, which can be noisy, and because then it is not taken into account that the system is time invariant.

Ca$\sim$En deals with the former problem, the noise, by means of the filter. The results obtained by MIS using longer windows are similar to the ones obtained by Ca$\sim$En with the filter. To

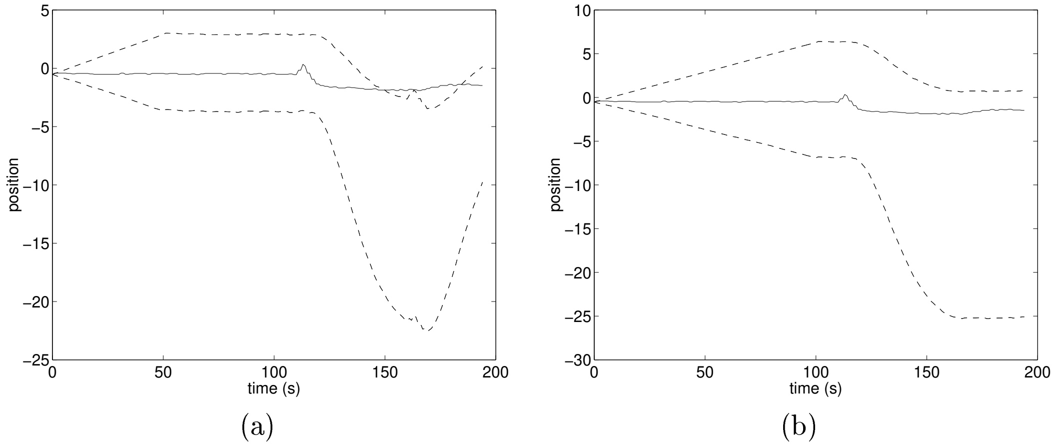


Figure 7-11: Scenario 40: envelopes for $TSNZ$ using windows of length (a) 50 s and (b) 100 s.

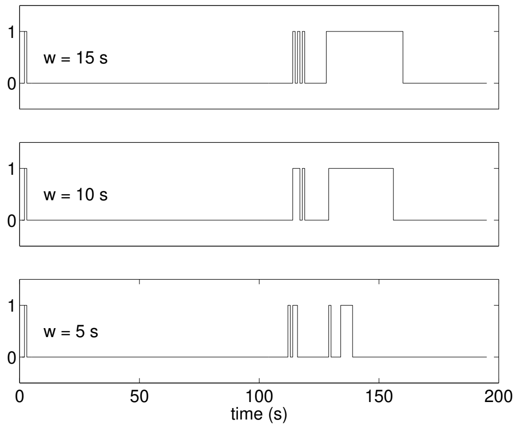


Figure 7-12: Scenario 40: indications of fault using the closed loop model and windows of length 5, 10 and 15 s.

obtain these results using Ca∼En, the parameter of the filter had to be adjusted empirically. Now, using MIS, the length of the window has to be adjusted.

The latter problem, the time invariance of the system, is not taken into account by Ca∼En.

In scenario 40 the fault is not detected with long windows. The interpretation of those results is not the goal of this work. Anyway, some possible interpretations are:

- There are reasons to avoid the use of too short windows. In a similar way, there can exist reasons to avoid the use of too long windows, at least to detect faults of short length like the one of scenario 40. The different examples show that the time to detect a fault depends on the length of the window. The longer is the window, the longer is the detection time. If the duration of the fault is shorter than this time, it is not detected. This is a question that may be studied in the future.

- The model of the system is too imprecise, i.e. the intervals are too wide. MIS is a useful tool in this case, as it can be used to obtain tighter interval models using them in an iterative procedure: adjustment of the interval parameters, simulation in different scenarios, new adjustment according to the simulation results, and so on.

In both cases, MIS is very useful, either to help to choose the window length or to help to adjust the interval model.

The results using the open loop model and the closed loop one are similar in all the scenarios. As the first model does not include the controller and the second one does it, it can be concluded that the controller is not faulty in any of the scenarios.

7.3 Two tanks benchmark

7.3.1 Introduction

This example comes from the project "Methodes qualitatives pour la commande supervisée et le diagnostic". This is the project number 98107 in the framework of the PROCOPE programme and involves a French team of LEA-SICA headed by Dr. Louise Travé-Massuyès (LAAS-CNRS, Toulouse) and a German team headed by Dr. Jan Lunze (TUHH, Hamburg). In order to compare the approaches of both teams to different problems of control and supervision, a benchmark based on a two tank system is used [59, 60].

This system is non-linear and consists of two interconnected tanks of circular cross-section. It is shown in figure 7-13. There is a pump that provides the incoming flow $q_{pump}(t)$. For the simulation of blockages and leaks in the tanks, various valves are installed.

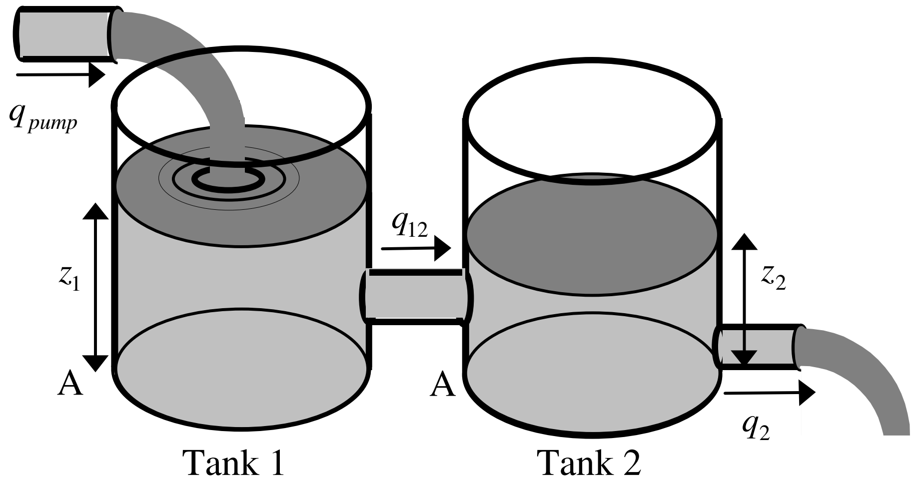


Figure 7-13: Two tanks system

7.3.2 Model

The dynamic model of the system in normal situation is derived using the incoming and outgoing mass flow and is described by the following difference equations:

- Tank 1:

$$A \cdot \frac{z_1(n) - z_1(n-1)}{T} = q_{pump}(n-1) - q_{12}(n-1) \tag{7.3}$$

- Tank 2:

$$A \cdot \frac{z_2(n) - z_2(n-1)}{T} = q_{12}(n-1) - q_2(n-1) \tag{7.4}$$

The values of the parameters of the model are the ones used in the PROCOPE project:

- T is the sampling time.
- n indicates the time point.
- A is the section of both tanks:

$$A = 0.0154 \quad \mathrm{m}^2 \tag{7.5}$$

- z_i is the level of the liquid in the tank i.
- q_{pump} is the incoming flow provided by the pump:

$$q_{pump}(n) = k_p \cdot u(n) \tag{7.6}$$

where

$$k_p = c_p \cdot A \tag{7.7}$$

being

$$c_p = 1.1 \cdot 10^{-2} \ \frac{\mathrm{m}}{\mathrm{s}} \tag{7.8}$$

and u indicates the state of the pump (off or on):

$$u = \{0, 1\} \tag{7.9}$$

- q_{12} is the flow from tank 1 to tank 2:

$$q_{12}(n) = \begin{cases} k_{12}\sqrt{z_1(n) - z_2(n)} & \text{if } z_1(n) > z_2(n) \\ -k_{12}\sqrt{z_2(n) - z_1(n)} & \text{if } z_2(n) > z_1(n) \end{cases} \tag{7.10}$$

where

$$k_{12} = c_{12} \cdot A \tag{7.11}$$

being

$$c_{12} = 5.52 \cdot 10^{-3} \ \frac{\sqrt{\mathrm{m}}}{\mathrm{s}} \tag{7.12}$$

- q_2 is the outflow:

$$q_2(n) = k_2\sqrt{z_2(n)} \tag{7.13}$$

where

$$k_2 = c_2 \cdot A \tag{7.14}$$

being

$$c_2 = 6.37 \cdot 10^{-3} \ \frac{\sqrt{\mathrm{m}}}{\mathrm{s}} \tag{7.15}$$

There can exist additional mass flows caused by leaks or blockages in the various tanks or pipes. They will be considered to simulate different faulty behaviours.

Therefore, the non-linear model of the system is:

- Tank 1:

$$z_1(n) = \begin{cases} z_1(n-1) + c_p \cdot T \cdot u(n-1) - c_{12} \cdot T \cdot \sqrt{z_1(n-1) - z_2(n-1)} \\ \qquad , \text{if } z_1(n-1) > z_2(n-1) \\ z_1(n-1) + c_p \cdot T \cdot u(n-1) + c_{12} \cdot T \cdot \sqrt{z_2(n-1) - z_1(n-1)} \\ \qquad , \text{if } z_2(n-1) > z_1(n-1) \end{cases} \tag{7.16}$$

- Tank 2:

$$z_2(n) = \begin{cases} z_2(n-1) + c_{12} \cdot T \cdot \sqrt{z_1(n-1) - z_2(n-1)} - c_2 \cdot T \cdot \sqrt{z_2(n-1)} \\ \qquad , \text{if } z_1(n-1) > z_2(n-1) \\ z_2(n-1) - c_{12} \cdot T \cdot \sqrt{z_2(n-1) - z_1(n-1)} - c_2 \cdot T \cdot \sqrt{z_2(n-1)} \\ \qquad , \text{if } z_2(n-1) > z_1(n-1) \end{cases} \tag{7.17}$$

7.3.3 Data

Input

The considered input sequence is indicated in table 7.1 and represented in figure 7-14.

Measurements

There are several available measurements. They are generated using the model of the system and the following values for the initial levels of the tanks: $z_{10} = 0.459$ m and $z_{20} = 0.159$ m.

It is considered that there are uncertainties in the measures:

time interval (s)	input u
$[0, 49]$	1
$[50, 59]$	0
$[60, 79]$	1
$[80, 119]$	0
$[120, 149]$	1
$[150, 189]$	0
$[190, 199]$	1
$[200, 249]$	0
$[250, 299]$	1
$[300, \infty]$	0

Table 7.1: Input of the two tanks system.

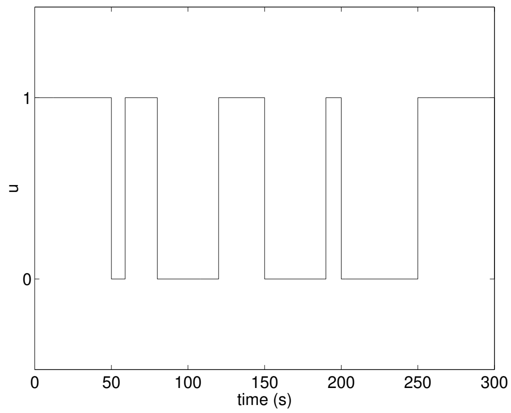


Figure 7-14: Input of the two tanks system.

- Level of the tanks z_i. They can range from 0.059 m to 0.659 m.

- Outflow q_2. As a consequence of the range of z_1, q_2 can range from 0 to $7.9635 \cdot 10^{-5}\ \frac{\mathrm{m}^3}{\mathrm{s}}$.

There is some uncertainty in the initial levels of the tanks. It is represented by the following intervals:

- Tank 1:

$$z_{10} = [0.409, 0.509]\ \mathrm{m} \tag{7.18}$$

- Tank 2:

$$z_{20} = [0.109, 0.209]\ \mathrm{m} \tag{7.19}$$

There are not interval parameters in the model of the two tanks system, so the only interval parameters are in the initial state.

Tests

Several tests have to be made in order to consider the following cases:

1. No fault.

2. Leakage in tank 1.

3. Blockage between tank 1 and 2.

4. Blockage between tank 1 and 2 and leakage in tank 1.

7.3.4 Related work

This benchmark is used by several people working on FD [21, 60]. The comparison between the different approaches can be made using the results and the computational effort needed to obtain them.

One of the approaches is the methodology that is being developed by the German team of the PROCOPE project. This FD system obtains a kind of envelopes in the qualitative parameter space and performs not only FD but also diagnosis.

7.3.5 Fault detection results

The main difficulties of this example are:

- It is not a SISO system: the levels at both tanks have to be taken into account to calculate the outflow.
- There are non-linearities in the model:
  - There are two models depending on whether the level of tank 1 is higher than the level of tank 2 or not.
  - The equations are non-linear due to the presence of square roots, which are non-linear functions.
  - The saturations of the tanks: there is a maximum level and a minimum one in each tank.

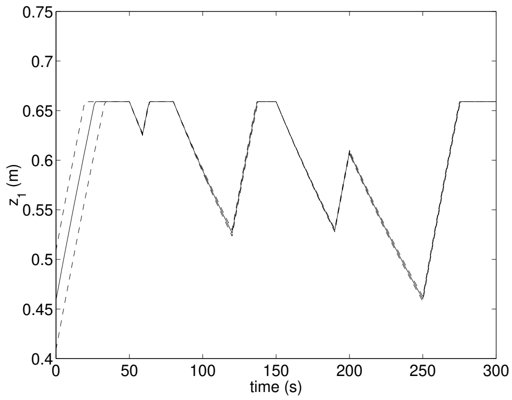
Due to these original characteristics of the system, MIS has had to be adapted to deal with this system. This section shows some results obtained using the adapted MIS for the different tests. In all tests a window length of ∞ is used.

No fault

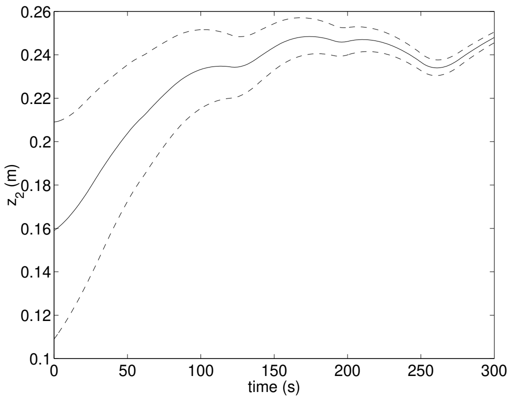
In this test the system is behaving normally, i.e. without faults.

The obtained envelopes for z_1, z_2 and q_2 are represented in figures 7-15 and 7-16. Figure 7-15a shows that the envelopes are very narrow from the point where they reach the maximum level of tank 1 onwards. At that point the uncertainty on the level of tank 1 disappears and after that point there is some uncertainty on that level due only to the uncertainty on the level of tank 2. There are not other sources of uncertainty because the model is a quantitative one. The uncertainties on the level and the outflow of tank 2 decrease with time because the only source of uncertainty is the initial state and its influence decreases with time. This is shown in figures 7-15b and 7-16. There are also represented in these figures the corresponding measured variables.

In this case there are not indications of faults, as it is shown in figure 7-17. This is the expected result.



(a)



(b)

Figure 7-15: Not faulty tanks: measurements and envelopes for the levels of (a) tank 1 and (b) tank 2

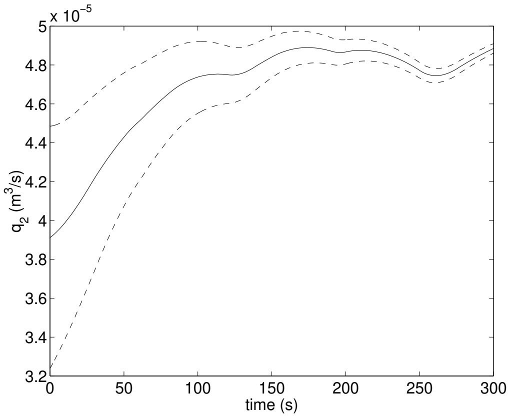


Figure 7-16: Not faulty tanks: measurements and envelopes for the outflow of tank 2.

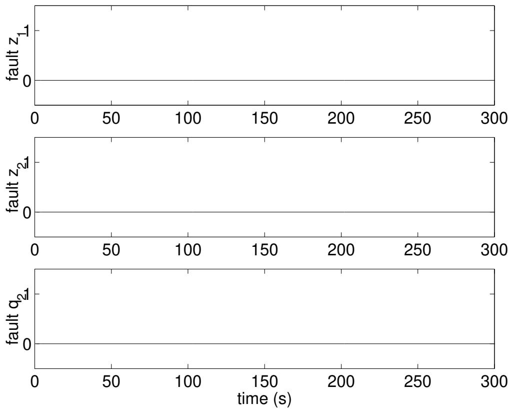


Figure 7-17: Not faulty tanks: indications of fault

Leakage

In this test there is a leakage in tank 1, so now the system is faulty. The additional flow in tank 1 is:

$$q_1(n) = k_{leak}\sqrt{z_1(n)} \tag{7.20}$$

where

$$k_{leak} = c_{leak} \cdot A \tag{7.21}$$

being

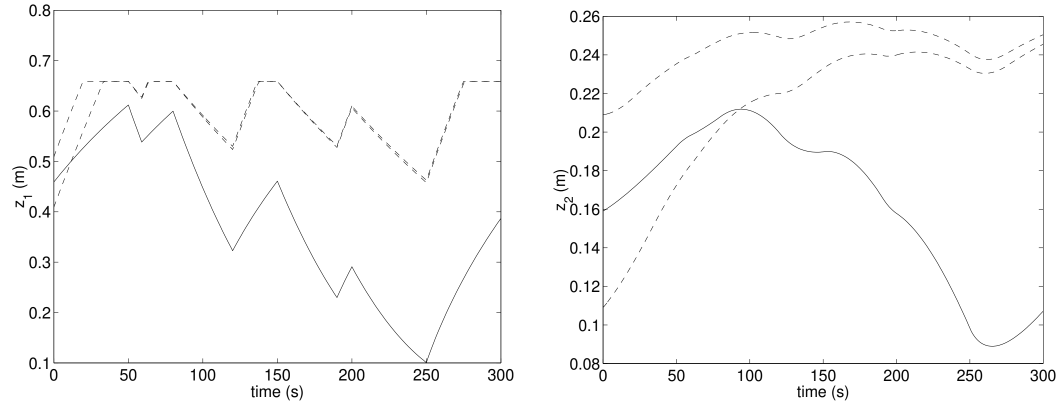
$$c_{leak} = 6.28 \cdot 10^{-3}\ \frac{\sqrt{\mathrm{m}}}{\mathrm{s}} \tag{7.22}$$

Therefore, now the model of tank 1 is:

$$z_1(n) = \begin{cases} z_1(n-1) + c_p \cdot T \cdot u(n-1) - c_{12} \cdot T \cdot \sqrt{z_1(n-1) - z_2(n-1)} - c_{leak} \cdot \sqrt{z_1(n-1)} \\ \qquad , \text{if } z_1(n-1) > z_2(n-1) \\ z_1(n-1) + c_p \cdot T \cdot u(n-1) + c_{12} \cdot T \cdot \sqrt{z_2(n-1) - z_1(n-1)} - c_{leak} \cdot \sqrt{z_1(n-1)} \\ \qquad , \text{if } z_2(n-1) > z_1(n-1) \end{cases} \tag{7.23}$$

This leakage causes that all variables are lower than expected (see figure 7-18). The outflow of tank 2 is not represented because it is very similar to the level of tank 2.

The fault is clearly detected by the envelopes of z_1, z_2 and q_2, as it can be seen in figure



(a) (b)

Figure 7-18: Tanks with leakage: measurements and envelopes for the levels of (a) tank 1 and (b) tank 2.

7-19.

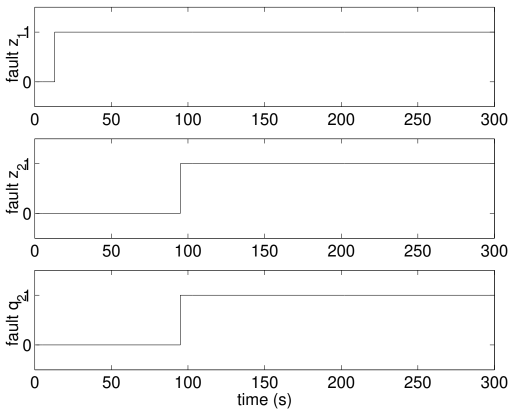


Figure 7-19: Tanks with leakage: indications of fault.

Some time is necessary to detect the fault when long windows are used because of the uncertainty on the initial state. This is clearly shown in figure 7-18b for z_2.

Blockage

There is now a blockage in the valve between tank 1 and tank 2. The consequences are that tank 1 gets full (figure 7-20a), tank 2 gets empty (figure 7-20b) and hence outflow becomes 0.

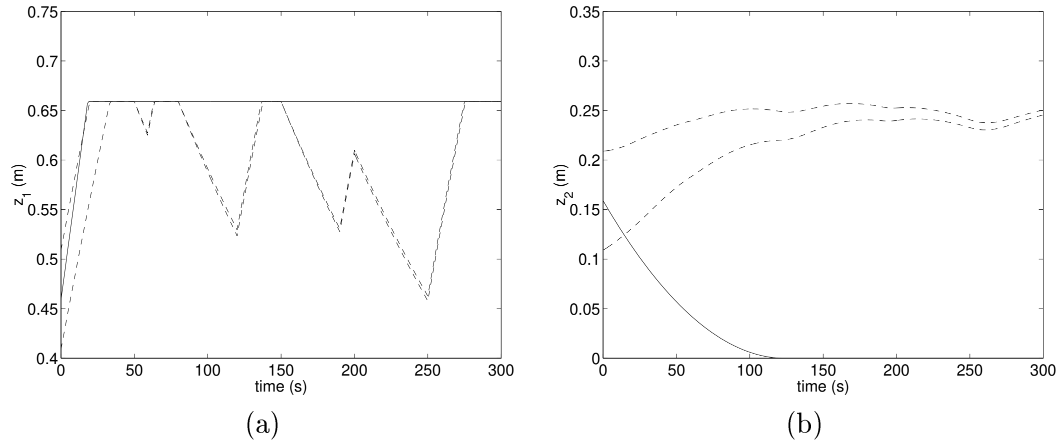


Figure 7-20: Tanks with blockage: measurements and envelopes for the levels of (a) tank 1 and (b) tank 2.

The different indications of fault are represented in figure 7-21.

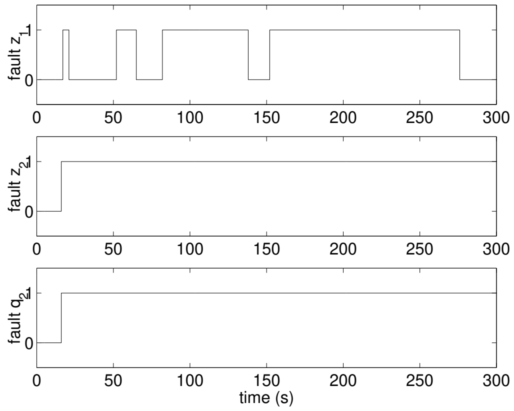


Figure 7-21: Tanks with blockage: indications of fault.

Blockage and leakage

Both previous faults happen in this case. The behaviour of z_1 is similar to the case when only the leakage occurs (see figure 7-22a). The indication of fault is represented in figure 7-23. On the other hand, z_2 and q_2 behave exactly the same in this case and when only the blockage occurs (see figure 7-22b). Therefore, the indication of fault, represented in figure 7-23, is the same as well.

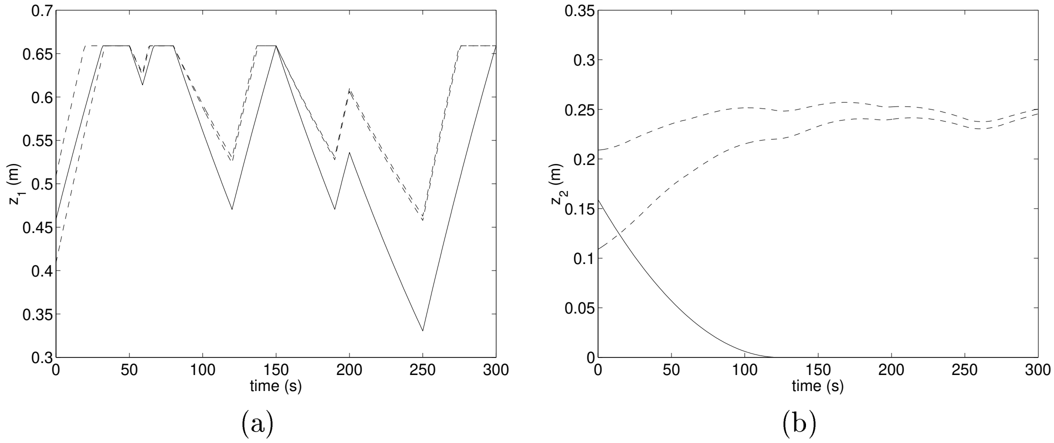


Figure 7-22: Tanks with blockage and leakage: measurements and envelopes for the levels of (a) tank 1 and (b) tank 2.

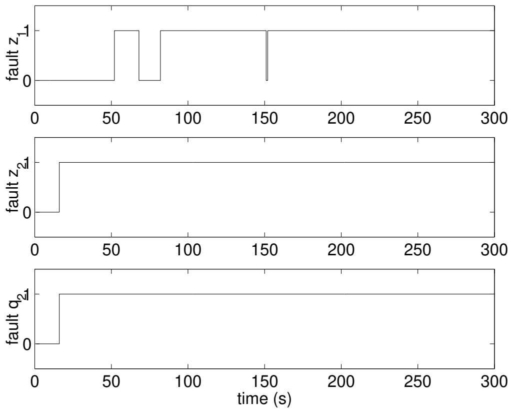


Figure 7-23: Tanks with blockage and leakage: indications of fault.

Summary

MIS has been adapted to this system due to its special characteristics. Some of these particularities will be extended in the future to allow the use of MIS for a wider kind of models.

This example shows that there are not false alarms and faults are detected clearly.

One of the goals of the PROCOPE project is to define a benchmark allowing to compare different techniques. MIS is a useful tool that allows to extract some conclusions and hence helps to redefine the benchmark. These conclusions are:

- The model is not an interval one.
- The cases that have been considered are not faults. They are better defined as failures [21], as they are abrupt changes in the behaviour of the system.
- Some uncertainty in the measurements, for instance noise, should be considered.

The obtained results can be compared with some results obtained by the German team of the PROCOPE project that are still unpublished. On one hand, the envelopes obtained by the German team are overbounded and wider than the ones shown here due to the use of qualitative labels. At this point, it seems that with MIS the results are better and are obtained with less computational effort. Nevertheless, it is worth noticing that MIS only detects faults and the German team performs FD and diagnosis.

7.4 Summary

This chapter has presented the application of MIS to several examples. They have been used to develop, test and validate them.

The first example is based on a real system and uses real data. It is the 2nd stage nozzles system of a gas turbine that was used in the Esprit project TIGER, which, as it has been stated in section 1.1, was the motivation of this work.

The window length used by Ca$\sim$En is 1 s. As the results using this window length are highly dependent on the measurements, which are noisy, a filter is used. The parameter of this filter is empirically adjusted. With MIS, similar results are obtained using longer windows and without filters. Now the length of the window has to be adjusted. MIS helps the user to choose the adequate window length allowing the use of different window lengths simultaneously.

MIS also can be used to obtain better interval models in an iterative procedure.

The second example is also based on a real system. It is the two tanks benchmark that is being used in the PROCOPE project number 98107 to compare different approaches to FD and diagnosis. As the benchmark is not completely defined yet, this is one of the goals of the project, the results that have been presented are only partial. Nevertheless, these results will help to redefine the benchmark in order to allow comparisons.

Chapter 8

Summary, conclusions and future work

8.1 Introduction

This chapter presents a general summary of the work, making special emphasis on its original contributions. It also provides some conclusions and outlines directions for future work.

8.2 Summary

This work presents the problem of the simulation of the behaviour of dynamic systems with uncertain parameters. The approach used in this work is based on the use of interval models, which include a representation of the uncertainty, and envelopes. The properties of the envelopes are introduced. Then, an overview of the simulators that can be used to generate envelopes is made in order to identify the properties of their envelopes and hence some of their strengths and weaknesses. A description of some simulators is presented and the properties of the envelopes that each of them generate are outlined and compared. These simulators use many different techniques that range from the pure qualitative ones to the pure quantitative ones passing through the many semiqualitative methods that exist between these two ends.

The conclusions of this overview are used to design and develop new simulators for interval models. They are based on the range computation method with a new approach: the use of

MIA, which is also introduced. The advantages and disadvantages of these new simulators are discussed. These discussions allow to improve their features and add new ones. The common features of these simulators are:

- Simulation of interval models of SISO systems expressed with difference equations.
- The step size of the simulation is fixed, but the user can choose it at the beginning of the simulation.
- Based on MIA.
- Directed roundings have been implemented.
- Input can be a sequence of intervals.
- The systems are considered time invariant, so the envelopes at a time point are determined computing them from the starting point of the simulation.

Moreover, the particularities of each of them are:

- MIS_0.
  - Generates the exact envelope in some particular cases.
  - Applies optimal coercion theorem.
  - Monotonicity is studied deriving as many times as needed.
  - There can be at most one multi-incident non-monotonic variable.
  - The study with respect to this single non-monotonic variable is based on the splitting of its parameter space in monotonic subspaces.
  - It is launched from Matlab 5.1 [4], although it is implemented in Maple V release 4 [3]. Therefore it needs the Symbolic Math Toolbox of Matlab.
- MIS.
  - Generates error-bounded envelopes.
  - Based on a branch and bound algorithm.

- Applies coercion theorems.
- Monotonicity is studied only with first derivative.
- Options to stop the iterations:
  * The error between the error-bounded envelopes is smaller than a maximum fixed at the beginning of the simulation.
  * The measurement is outside the overbounded envelope or inside the underbounded envelope.
  * Both options above at the same time.
- Sliding time windows of different lengths can be used simultaneously.
- Options for the starting point of each window:
  * The error-bounded envelopes.
  * The measurement.
  * The error-bounded envelopes or the measurement depending on the location of the measurement with respect to the envelopes.
- Implemented using Matlab version 5.1 for Unix [4]. Symbolic computations are performed with Maple V release 4 [3] through the Symbolic Math Toolbox. Moreover, it uses C++ programs as MEX-files.

The implementation in Matlab facilitates their future integration into a supervision framework that is being developed based on Matlab and Simulink [87]. It is also planned to be used to improve the prediction and FD algorithms of the Ca∼En simulator [81].

The simulators are tested and validated applying them for FD in several examples, both academic and real. The problem of the faults and their detection and some of the approaches that have been used for this task are introduced. There are, among them, the analytical redundancy methods based on the model of the system, which include the approach used in this work. These methods compare the output of a system with a reference one generated through simulation and the faults are detected using the discrepancies. In this case the reference are the envelopes and a fault is detected when the measurement is outside the envelope. The importance of the properties of the envelopes related to FD is discussed.

The approach used in this work is based on the error-bounded envelopes. The fault is detected when the measurement is outside the overbounded envelope.

The only way to consider that the system is time invariant is simulating with respect to the starting point. This means that the computational effort increases at each step of the simulation. Sliding time windows can be used to avoid this.

When sliding time windows are used for FD, the envelopes at the end of a window can be computed either from the envelopes at the beginning of the window or from the measurements (which can be intervals) at that point. In the first case, there is a propagation of the uncertainty. In the second case, the envelopes are highly dependent on the measurements, which can be noisy. There is a third option, the one used by Ca$\sim$En, consisting in using the first option if the measurement is outside the overbounded envelope and the second one if it is not.

The systems are considered time invariant when sliding time windows are used. The variation speed depends on the length of the window. Hence, this length has an influence over the capability to detect some kinds of faults. To help the user to deal with the length of the window, MIS can use several window lengths simultaneously.

8.3 Original contributions of this work

To obtain the exact envelopes is a computationally unfeasible task. A new simulator that does it in very special cases has been developed. But it is not necessary to obtain the exact envelopes when they are to be used for FD. It has been shown that the same results can be obtained using error-bounded envelopes, i.e. a pair of envelopes, an overbounded one an underbounded one. The computation of these envelopes is simpler and needs less computations. Hence, new simulators that compute error-bounded envelopes are introduced and their properties are discussed.

If the error-bounded envelopes are computed starting form the initial state, the computation effort increases at each simulation step. This is solved using sliding time windows. Then there is a relation between the computation effort and the characteristics of the results: when the window length increases the computation effort is lower and the amount of false alarms and missed alarms is lower as well. MIS helps the user to choose the window length by allowing to simulate using different window lengths simultaneously.

The main advantage of the use of sliding time windows is that it allows the use of the error-bounded envelopes in real time for on-line FD.

Therefore, the contributions of this work that are original are:

- A comparative analysis of some existing envelope generators with regard to the properties of the envelopes they generate.
- The definition of error-bounded envelopes.
- The design and development of different simulators that generate error-bounded envelopes.
- The use of an implementation of MIA, which uses directed roundings to maintain the semantics, to generate the error-bounded envelopes.
- The use of multiple sliding time windows.
- The implementation of these simulators in Matlab to facilitate its future integration into a supervision framework that is being developed based on Matlab and Simulink [87].
- The application of these simulators to FD in academic and real examples.

8.4 Conclusions

From the overview on the simulators that can generate envelopes, it can be concluded that many authors of simulators did not try to assess the properties of the envelopes for their simulators. These properties are very important for FD as they indicate the possibility to have missed alarms and false alarms. In some of the cases in which the properties are not assessed, they can be deduced studying the algorithms used to simulate, but in other occasions the properties remain unknown.

When the properties of the envelopes are known, it has been seen that there are some simulators that generate overbounded envelopes, so they will miss alarms, and other simulators that generate underbounded envelopes, so they will indicate false alarms. Moreover, there are simulators that generate envelopes that are neither underbounded nor overbounded because they are neither complete nor sound. These simulators will result in missed alarms and false alarms.

The results are highly dependent on the goodness of the model and the measurements.

Modelling and simulation are interdependent tasks. To test if a model is good, a good simulator is needed and to have a good simulation, a good model is needed. MIS allows to refine the modelling process by providing guaranteed simulation results. For instance, if the model of the system is too imprecise (resp. too precise), i.e. the intervals are too wide (resp. tight), the envelopes also will be too wide (resp. tight) and there will be missed (resp. false) alarms. MIS is then a useful tool, as it can be used to obtain better interval models adjusting the interval parameters in an iterative procedure.

The use of sliding time windows is necessary to perform FD on-line. The problem is to choose the length of the window, which has consequences on the computation effort needed and the characteristics of the results of the FD. This can be seen when the measurements are noisy, as false alarms can be generated due to the noise of the sensors. This problem can be dealt with by several ways:

- Including in the interval model the uncertainty of the measurements.
- Using interval measurements, which include their uncertainty, to compare the measurements and the envelopes. This matter is under study according to the point of view of modal intervals.

The use of different window lengths and the comparison of the results obtained by each of them help the user to choose the window length. This can be done with MIS, which allows to use several window lengths simultaneously.

Therefore, the tools presented in this work help the user to set up a FD system based on error-bounded envelopes by providing means to adjust the model and to fix the length of the sliding time windows.

8.5 Future work

This section enumerates some questions that are still open after this work or that have been originated by it.

It has been shown with several examples that it is no good to use too short windows because the results are highly dependent on the last measurements, which can be noisy, and because then it is not taken into account that the system is time invariant. In a similar way, it can be studied if there exist reasons to avoid the use of too long windows, at least to detect faults of short length (see section 7.2.5).

The measurement at a time point is a real number and has an uncertainty associated with it, so it should be better expressed with an interval. This should to be taken into account when this measurement is compared with the error-bounded envelopes to detect faults. This implies to perform a comparison between the envelopes and an interval and hence the possible answers are not only "the measurement is inside of the underbounded envelope" or "the measurement is outside of the overbounded envelope". The semantic implication of these answers have to be studied under the optics of modal intervals.

As it has been shown, a very interesting feature of modal intervals is the semantics. A future work is to study whether envelopes with different semantics can be used not only to detect the faults but also to diagnose the faulty parameter. For instance, if a system has two physical parameters a and b, envelopes with the semantics "for every a there exists b so that..." or "for every b there exists a so that..." can be obtained. If a system is faulty and its output belongs to only one of these two envelopes, it should be possible to determine whether it is a or b that is faulty.

This future work is based on the use of new algorithms that are under development and that overcome some of the limitations of the current algorithms:

- The current algorithms are limited to SISO systems. New algorithms will deal with MIMO (Multiple Input, Multiple Output) systems. This is a requirement of, for instance, the two tanks benchmark (see section 7.3), which is a system with two outputs. In this case, however, an adaptation to the particular model has been done.

- A similar question is the adaptation of the algorithms to deal with non-linearities: non-linear models, saturations, etc. The necessary adaptation in the case of the two tanks benchmark is particular.

In [15] it is claimed that it is not necessary to study the evolution of all the points belonging to an uncertainty region in order to know the evolution of the region. The study of the evolution of the points belonging to its surface is enough. The possible application of this to modal interval simulation gives another direction for research.

The user-friendliness of the simulators has to be increased to facilitate their use. One step in that direction is their implementation in Simulink, which is under development. Some of the consequences of this implementation will be an improved flexibility of the simulators and the possibility to combine several simulators to detect faults in complex systems. This will allow to use them, with a minimum of difficulties, in new applications in which now it is difficult or even impossible.

This task is part of a more important one: the integration of these simulators into a supervision framework that is under development [87]. A similar task that is planned for the future is the integration of these simulation methods in the Ca$\sim$En simulator [81] to improve its prediction and FD algorithms.

Bibliography

[1] *Advanced Continuous Simulation Language (ACSL).* mailto:acsl-sales@aegisrc.com, http://aegisrc.com/ACSL/default.htm. AEgis Research Corporation.

[2] *Dymola.* mailto:info@dynasim.se, http://www.dynasim.se. Dynasim AB.

[3] *Maple V.* mailto:info@maplesoft.com, http://www.maplesoft.com. Waterloo Maple, Inc.

[4] *Matlab-Simulink.* mailto:info@mathworks.com, http://www.mathworks.com. The Math Works, Inc.

[5] *MicroSaint.* http://www.madboulder.com. Micro Analysis and Design, Inc.

[6] *Omola.* http://www.control.lth.se/∼cace/omsim.html. Department of Automatic Control. Lund Institute of Technology.

[7] *Shift.* http://www.path.berkeley.edu/shift. California Partners for Advanced Transit and Highways (PATH) . University of California at Berkeley.

[8] *Simulation Program with Integrated Circuit Emphasis (PSPICE).* mailto:info@microsim.com, http://www.microsim.com. MicroSim Corporation.

[9] *Witness.* mailto:info@lanner.com, http://www.lanner.com. Lanner Group, Ltd.

[10] G. Alefeld and J. Herzberger. *Introduction to interval computations.* Academic Press, 1983.

[11] J. Armengol, L. Travé-Massuyès, J. Vehí, and J. L. de La Rosa. A survey on interval model simulators and their properties related to fault detection. *14th IFAC World Congress 1999. Beijing, China*, O:511–519, 1999.

[12] D. N. Arnold. Two disasters caused by computer arithmetic errors. *http://www.math.psu.edu/dna/455.f96/disasters.html*, 1996.

[13] B. Bennett. *Simulation fundamentals.* Prentice Hall, 1995.

[14] D. Berleant and B. Kuipers. Qualitative and numerical simulation with q3. *Recent advances in qualitative physics*, 1992.

[15] A. Bonarini and G. Bontempi. A qualitative simulation approach for fuzzy dynamical models. *ACM Transactions on Modeling and Computer Simulation*, 4(4):258–313, 1994.

[16] G. Bontempi. Qua.si. iii: A software tool for simulation of fuzzy dynamical systems. *Proceedings of the European Simulation Multiconference ESM 96. Ghent, Belgium*, pages 615–619, 1996.

[17] K. Bousson and L. Travé-Massuyès. Putting more numbers in the qualitative simulator ca-en. *2nd International Conference on Intelligent Systems Engineering ISE 94. Hamburg-Harburg, Germany*, pages 62–69, 1994.

[18] P. Bratley, L. Bennett, and L. Schrage. *A guide to simulation.* Springer-Verlag, 1987.

[19] G. Cargo and O. Shisha. The bernstein form of a polynomial. *Journal of Res. Nat. Bur. Standards Sect. B*, 70B:79–81, 1966.

[20] F. Cellier. *Continuous system modeling.* Springer-Verlag, 1991.

[21] J. Chen and R. Patton. *Robust model-based fault diagnosis for dynamic systems.* Kluwer, 1998.

[22] L. Chittaro, G. Guida, C. Tasso, and E. Toppano. Functional and teleological knowledge in the multi-modeling approach for reasoning about physical systems: a case study in diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1718–1751, 1993.

[23] G. Coghill. *Mycroft: a framework for constraint based fuzzy qualitative reasoning.* PhD thesis, Heriot-Watt University, Edinburgh, Scotland, 1996.

[24] A. Connell and R. Corless. An experimental interval arithmetic package in maple. *Interval Computations*, 1993.

[25] G. Corliss. Intpak for interval arithmetic in maple: introduction and applications. *Journal of Symbolic Computation*, 11, 1994.

[26] P. Danès. *Interface symbolique-numerique dans la simulation qualitative des systèmes dynamiques.* PhD thesis, Université Paul Sabatier, Toulouse, France, 1995.

[27] L. J. De Miguel and J. R. Perán. Métodos de detección y diagnóstico de fallos basados en el modelo de la planta. *Informática y Automática*, 29(1):3–29, 1996.

[28] K. D. Forbus. Commonsense physics: a review. *Annual Review of Computer Science*, 3:197–232, 1988.

[29] E. Gardeñes and H. Mielgo. Modal intervals: functions. *Polish Symposium on Interval and Fuzzy Mathematics. Poznan, Poland*, 1986.

[30] E. Gardeñes, H. Mielgo, and M. A. Sainz. Presentation of the research group sigla/x. Technical report, Universitat de Girona. Catalonia, Spain., 1995.

[31] E. Gardeñes, H. Mielgo, and A. Trepat. Modal intervals: reasons and ground semantics. *Lecture Notes in Computer Science*, 212:27–35, 1986.

[32] J. Garloff. Convergent bounds for the range of multivariate polynomials. *Interval Mathematics 1985*, pages 37–56, 1985.

[33] J. Garloff. The bernstein algorithm. *Interval Computations*, 2:154–168, 1993.

[34] R. M. Gasca. *Razonamiento y simulación en sistemas que integran conocimiento cualitativo y cuantitativo.* PhD thesis, Universidad de Sevilla, Spain, 1998.

[35] R. M. Gasca, J. A. Ortega, and M. Toro. Simulación semicualitativa. *Jornades Hispano-Franceses de Sistemes Intel·ligents i Control Avançat. Barcelona, Catalonia, Spain*, 1996.

[36] E. Hansen. *Global optimization using interval analysis.* Marcel Dekker, 1992.

[37] E. Hyvönen. Constraint reasoning based on interval arithmetic: the tolerance propagation approach. *Artificial Intelligence*, 58:71–112, 1992.

[38] E. Hyvönen. Evaluation of cascaded interval function constraints. *International Workshop on Constraint-Based Reasoning*, 1995.

[39] E. Hyvönen and S. De Pascale. In c++ library for interval computations. *Reliable Computing. Proceedings of Application of Interval Computations*, pages 85–90, 1995.

[40] E. Hyvönen and S. De Pascale. *Interval computations on the spreadsheet*, pages 169–210. Kluwer, 1996.

[41] E. Hyvönen and S. De Pascale. Range solver for microsoft excel. user's guide. Technical report, VTT Information Technology. Espoo, Finland, 1996.

[42] E. Hyvönen and S. De Pascale. Range solver: Next generation spreadsheet computing for microsoft excel. Technical report, VTT Information Technology. Espoo, Finland, 1996.

[43] E. Hyvönen and S. de Pascale. A new basis for spreadsheet computing: Interval solver for microsoft excel. *Proceedings of Innovative Applications of Artificial Intelligence (IAAI 99), Menlo Park, California, USA*, 1999.

[44] R. Isermann and P. Ballé. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5:707–719, 1997.

[45] A. Jorba Monte and J. Masdemont Soler. *Introducció a la simulació.* Edicions UPC, 1995.

[46] W. Kahan. Lecture notes on the status of ieee standard 754 for binary floating point arithmetic. 1996.

[47] H. Kay. Numerical behavior envelopes for qualitative models. *6th International Workshop on Qualitative Reasoning about Physical Systems QR 92*, pages 252–267, 1992.

[48] H. Kay. Semiquantitative simulation: successes, failures, and future directions. 1995.

[49] H. Kay. *Refining imprecise models and their behaviours.* PhD thesis, University of Texas at Austin, 1996.

[50] R. Kearfott. *Rigorous global search: continuous problems.* Kluwer Academic Publishers, 1996.

[51] R. Kearfott, W. Dawande, K. Du, and H. C.Y. Intlib: a portable fortran 77 elementary function library. *Interval Computations*, 3(5):96–105, 1992.

[52] R. Kearfott and V. Kreinovich. *Applications of interval computations.* Kluwer, 1996.

[53] U. Keller, T. Wyatt, and R. Leitch. Frensi - a fuzzy qualitative simulation method. *Workshop on Applications of Interval Analysis to Systems and Control with special emphasis on recent advances in Modal Interval Analysis (MISC 99). Girona, Catalonia, Spain*, pages 305–313, 1999.

[54] O. Knuppel. Profil/bias - a fast interval library. *Computing*, 53:277–287, 1993.

[55] B. Kuipers and D. Berleant. Using incomplete quantitative knowledge in qualitative reasoning. *AAAI 88*, pages 324–329, 1988.

[56] B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[57] A. M. Law and W. D. Kelton. *Simulation, modelling and analysis.* McGraw-Hill, 1991.

[58] R. Lohner. *Einschliessung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben.* PhD thesis, Universität Karlsruhe, Germany, 1988.

[59] J. Lunze, B. Nixdorf, and H. Richter. Hybrid modelling of continuous-variable systems with application to supervisory control. *Journal of Process Control*, 1998.

[60] J. Lunze, B. Nixdorf, and J. Schröder. Deterministic discrete-event representations of linear continuous-variable systems. *Automatica*, 35(3):395–406, 1999.

[61] S. Markov and R. Angelov. An interval method for systems of ode. *Lecture Notes in Computer Science*, 212:103–108, 1985.

[62] R. Milne, C. Nicol, M. Ghallab, L. Travé-Massuyès, K. Bousson, C. Dousson, J. Quevedo, J. Aguilar, and A. Guasch. Tiger: real-time situation assessment of dynamic systems. *Intelligent Systems Engineering, Vol. 3, No. 3*, pages 103–124, 1994.

[63] A. Missier. *Structures mathematiques pour le calcul qualitatif - Contribution a la simulation qualitative.* PhD thesis, INSA, Toulouse, France, 1991.

[64] MODIFY. *Activity Fault Diagnosis. Final Report.* TEMPUS_JEP 07759-94 - MODIFY. Upgrading the Hungarian Higher Education in System Modeling, Fault Diagnosis and Fuzzy Logic, 1997.

[65] J. Montmain. From diapason research program to its industrial application in nuclear fuel reprocessing. *SAFEPROCESS 97*, pages 209–216, 1997.

[66] R. E. Moore. *Interval arithmetic and automatic error analysis in digital computing.* PhD thesis, Stanford University, USA, 1962.

[67] R. E. Moore. *Interval analysis.* Prentice-Hall, 1966.

[68] R. E. Moore. *Methods and applications of interval analysis.* Studies in Applied Mathematics (SIAM), 1979.

[69] A. J. Morgan. *Qualitative behaviour of dynamic physical systems.* PhD thesis, University of Cambridge, 1988.

[70] J. Quevedo, J. Armengol, and M. Vassilaki. Gas turbine fault detection based on model simulation of the process and its controllers. *Proceedings of the European Simulation Multiconference (ESM 94). Barcelona, Catalonia, Spain.*, pages 260–264, 1994.

[71] H. Ratscheck and J. Rokne. *Computer methods for the range of functions.* Ellis Horwood, 1984.

[72] H. Ratschek and J. Rokne. *New computer methods for global optimization.* Ellis Horwood, 1988.

[73] J. Rokne. Bounds for an interval polynomial. *Computing*, 18:225–240, 1977.

[74] J. Saludes, V. Puig, and J. Quevedo. Determination of window length for a new algorithm in adaptive threshold generation. *Symposium on Qualitative System Modeling, Qualitative Fault Diagnosis and Fuzzy Logic and Control. Budapest, Hungary*, 1997.

[75] C. Schnepper and M. Stadtherr. Robust process simulation using interval methods. *Computers Chem. Eng.*, 20:187–199, 1996.

[76] Q. Shen and R. Leitch. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man an Cybernetics*, 23(4):1038–1061, 1993.

[77] SIGLA/X. Modal intervals. *Applications of Interval Analysis to Systems and Control. Proceedings of MISC 99*, pages 157–227, 1999.

[78] V. Stahl. Interval methods for bounding the range of multivariate polynomials. Technical report, Research Institute for Symbolic Computation. Johannes Kepler University. Linz, Austria, 1995.

[79] V. Stahl. Interval methods for bounding the range of univariate polynomials. Technical report, Research Institute for Symbolic Computation. Johannes Kepler University. Linz, Austria, 1995.

[80] P. Struss. Problems of interval-based qualitative reasoning. *Readings in qualitative reasoning about physical systems*, pages 288–305, 1990.

[81] L. Travé-Massuyès, P. Dague, and F. Guerrin. *Le raisonnement qualitatif pour les sciences de l'ingénieur.* Editions Hermes, 1997.

[82] L. Travé-Massuyès and R. Milne. Diagnosis of dynamic systems based on explicit and implicit behavioural models: an application to gas turbines in esprit poject tiger. *Proceedings of Scandinavian Conference on Artificial Intelligence (SCAI 95)*, pages 169–183, 1995.

[83] L. Travé-Massuyès and R. Milne. Tiger: gas turbine condition monitoring using qualitative model based diagnosis. *IEEE Expert Intelligent Systems and Applications*, 1997.

[84] J. Tupper. *Graphing equations with Generalized Interval Arithmetic.* PhD thesis, University of Toronto. Canada, 1996.

[85] R. Van Iwaarden. *An inproved unconstrained global optimization algorithm.* PhD thesis, University of Colorado at Denver, USA, 1996.

[86] J. Vehí. *Anàlisi i disseny de controladors robustos mitjançant intervals modals.* PhD thesis, Universitat de Girona. Catalonia, Spain, 1998.

[87] J. Vehí, I. Ferrer, and J. Armengol. Using interval and symbolic computations to deal with uncertain dynamic systems. *1st NICONET workshop on numerical software in control engineering. València, Spain*, page 13, 1998.

[88] J. Vehí, J. Rodellar, M. A. Sainz, and J. Armengol. Analysis of the robustness of predictive controllers via modal intervals. *Reliable Computing*, To appear.

[89] M. Vescovi, A. Farquhar, and Y. Iwasaki. Numerical interval simulation. *International Joint Conference on Artificial Intelligence IJCAI 95. Montreal, Quebec, Canada*, pages 1806–1813, 1995.

[90] M. R. Vescovi. *La Répresentation des conaissances et le raisonnement sur les systèmes physiques.* PhD thesis, Université de Savoie, 1991.

[91] M. R. Vescovi and L. Travé-Massuyès. A constructive approach to qualitative fuzzy simulation. *Sixth International Workshop on Qualitative Reasoning about Physical Systems (QR 92). Edinburgh, Scotland*, pages 268–280, 1992.

[92] D. S. Weld and J. de Kleer. *Readings in qualitative reasoning about physical systems.* Morgan Kaufmann Publishers, 1990.

[93] M. Wiegand. *Constructive qualitative simulation of continuous dynamic systems.* PhD thesis, Heriot-Watt University, Edinburgh, Scotland, 1991.

[94] M. Wiegand and R. Leitch. A predictive engine for the qualitative simulation of dynamic systems. *4th International Conference on the Application of Artificial Intelligence in Engineering*, 1989.

References generated by this work

[I] J. Quevedo, J. Armengol, and M. Vassilaki. Gas turbine fault detection based on model simulation of the process and its controllers. *Proceedings of the European Simulation Multiconference (ESM 94), Barcelona, Catalonia, Spain*, pages 260–264, 1994.

[II] J. Armengol, L. Travé-Massuyès, J. Ll. de la Rosa, and J. Vehí. Envelope generation for interval systems. *Seminario sobre técnicas cualitativas. VII Congreso de la Asociación Española para la Inteligencia Artificial (CAEPIA 97), Málaga, Spain*, pages 33–48, 1997.

[III] J. Armengol. Simulation of dynamical systems with structured uncertainty: an overview. Technical report, Universitat de Girona, Catalonia, Spain, 1998.

[IV] J. Armengol, L. Travé-Massuyès, J. Vehí, and J. Ll. de la Rosa. On Modal Interval Analysis for envelope determination within the Ca$\sim$En qualitative simulator. *7th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference (IPMU 98), Paris, France*, pages 110–117, 1998.

[V] J. Armengol, L. Travé-Massuyès, J. Vehí, and M. Á. Sainz. Envelope generation using Modal Interval Analysis. Technical report, Universitat de Girona, Catalonia, Spain, 1998.

[VI] J. Armengol, L. Travé-Massuyès, J. Vehí, and M. Á. Sainz. Modal Interval Analysis for error-bounded semiqualitative simulation. *Butlletí de l'Associació Catalana d'Intel·ligència Artificial*, (14-15):223–231, 1998.

[VII] J. Vehí, I. Ferrer, and J. Armengol. Using interval and symbolic computations to deal with uncertain dynamic systems. *1st NICONET workshop on numerical software in control engineering, València, Spain*, page 13, 1998.

[VIII] J. Armengol, J. Vehí, L. Travé-Massuyès, and M. Á. Sainz. Generation of error-bounded envelopes by means of Modal Interval Analysis. *Applications of Interval Analysis to Systems and Control*, pages 271–281, 1999.

[IX] J. Armengol, L. Travé-Massuyès, J. Vehí, and M. Á. Sainz. Generation of error-bounded envelopes using Modal Interval Analysis. *10th International Workshop on Principles of Diagnosis (DX 99). Loch Awe, Scotland, UK*, 1999.

[X] J. Armengol, L. Travé-Massuyès, J. Vehí, and J. Ll. de la Rosa. A survey on interval model simulators and their properties related to fault detection. *14th IFAC World Congress 1999, Beijing, China*, 1999.

[XI] J. Armengol, L. Travé-Massuyès, J. Vehí, and M. Á. Sainz. Semiqualitative simulation using Modal Interval Analysis. *14th IFAC World Congress 1999, Beijing, China*, 1999.

[XII] J. Armengol, J. Vehí, L. Travé-Massuyès, and M. Á. Sainz. Application of modal intervals to the generation of error-bounded envelopes. *Reliable Computing*, Submitted.