DEVELOPMENT AND IMPLEMENTATION OF AN ALGORITHM FOR THE CALCULATION OF POSITIVE INVARIANT SETS USING INTERVAL METHODS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF MASTER OF SCIENCE IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2015

DANIEL LÓPEZ MADRID

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

Table of Contents

Word Count: 9,919

List of Figures

List of Tables

List of Acronyms

CSP – Constraint Satisfaction Problem

SIVIA – Set Inversion Via Interval Analysis

SMART – Second International Symposium on Set Membership - Applications, Reliability and Theory

Abstract

The study and research of the stability properties of dynamic systems has always been an important topic in control engineering. Most of the existing procedures are based on Lyapunov's Stability and are constructed around the concept of positive invariant sets. Nowadays, one of the most important problems is the inexistence of procedures to calculate these stability regions. Until now, they were defined based on the energy of the system and for complex systems it was difficult to find suitable solutions. The objective of this research is to develop an algorithm which can calculate these regions, in a guaranteed way, for any system starting from a simple initial intuition. The use of interval methods allows this algorithm to find solutions to this problem that otherwise would be impossible to obtain and to work with systems that present uncertainty. Finally, the algorithm is applied to a simple system like a pendulum to show its results and illustrate its capabilities.

Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

iv. Further information on the conditions under which disclosure, publication and commercialisation of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/display.aspx?DocID=487), in any relevant Dissertation restriction declarations deposited in the University Library, The University Library's regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University's Guidance for the Presentation of Dissertations.

Acknowledgements

First, I would like to thank my family for all the support and help I have received during all my life and especially since I have been studying away from home.

This work is dedicated to Claudia. Thank you for all the times you have made me laugh and for still being my source of joy.

I want to personally thank my tutor, Dr Alexandru Stancu, for all his support and supervision during this difficult journey.

Last but not least, I would like to express my gratitude to Professor Luc Jaulin from ENSTA Bretagne, France, for the guidance and valuable consultation that he provided me at all times.

Chapter 1

Introduction

1.1 Background and Motivation

The control engineering research community has always been involved in the study of the concept of stability [1]–[3]. Since Aleksandr Lyapunov introduced his work on this topic [4] and because of the importance of the stability properties of a system, researchers have developed numerous stability methods for both linear and nonlinear systems based on Lyapunov's work [5], [6].

Most control problems can be expressed in a set theoretic context. Set-theory can be used to characterize important properties in a control system design problem [7] such as uncertainties, constraints and design specifications. A key concept that links Lyapunov functions and set theory is positive invariance [8].

Consider a simple autonomous system defined by

$$\dot{x} = f(x(t)) \tag{1.1}$$

It can be assumed that this system's equations are defined in a proper open set $\mathbb{O}$ such that

$$\mathbb{O} \subseteq \mathbb{R}^n \tag{1.2}$$

Moreover, there exist a solution for all $t \geq 0$ for every initial condition $x(0) \in \mathbb{O}$. Then, a set $\mathbb{S} \subseteq \mathbb{O}$ is *positive invariant* to the studied system if all the system solutions with $x(0) \in \mathbb{S}$ are such that

$$\forall t > 0, x(t) \in \mathbb{S} \tag{1.3}$$

Lyapunov functions are positive definite functions normally derived from the system's energy dissipation (or preservation) and which have the property of decreasing with time. It can be seen in principle that the concept of positive invariance is not related with a Lyapunov function, but the idea of set invariance can be used to create a more general theory than the one developed by Lyapunov. For example, while a Lyapunov function is required to be positive definite, this is no longer necessary for positive invariant sets and it is useful for many problems in which the desired system do not need to have this property.

One of these methods based on Lyapunov's stability and set theory is V-Stability [9]. It has been formulated to solve the stability problem of time invariant systems and can be used together with interval methods for the same objective in the case of differential inclusions.

V-Stability is based on the idea that all the trajectories of a time invariant system are captured by a positive invariant set. Which means that all the trajectories go towards a region that once they enter they will always remain inside.

This method can be extended to time variant systems and is referred to as G-Stability [10]. Now, instead of finding a fixed region, both the system and the region can vary with time. In case of time variant systems, the region evolves into a time varying bubble denominated as a *tube*. If all the trajectories that enter the tube, remain always inside, then this tube is considered to be a capture tube. For a system to be G-stable, it must have a capture tube. Hence, in order to prove if a system is G-Stable we have to find a capture tube candidate.

The majority of the systems contain various sources of uncertainty, due to the imperfections in sensors or actuators, which cannot be neglected. Probabilistic methods assume that these uncertainties follow a certain distribution but sometimes these assumptions are not realistic. Moreover, these uncertainties make impossible to describe the system by a single state equation but need to be defined by a *differential inclusion* [9]. Interval is a very powerful tool to deal with differential inclusions. This technique allows us to work with ranges instead of individual values and to avoid the need for the uncertainties to follow a predefined distribution. Interval methods can be also used in case of complex systems to find the region which enclose the solution.

Another important problem for the control engineering community is how to prove that a group of autonomous robots can operate safely, i.e. no collision among them. It has been shown that this problem can be solved by calculating the correspondent capture tubes of the autonomous robots. Then, if the calculated capture tubes do not collide neither do the robots [10].

1.2 Aims of the Dissertation

Classical methods rely on studying the energy of the system to find a candidate stability region. This method is restricted to a small class of systems and difficult to implement for complex systems. To my knowledge and my colleagues', it does not exist a method to calculate, in a guaranteed way, these stability regions for non-linear differential inclusions. Hence, the aim of this research is to develop an algorithm capable of establishing the stability regions of a system based on a set of initial values and their future trajectories. Following is a brief description of the objectives of this dissertation:

- Use of interval methods and *contractor programming* [11] to achieve an inner and outer approximation of the solution set. This approximation can be used to do complex calculations and to find a guaranteed stable region. These tasks may be impossible by other means due to the complicated shape of the sets.

- Include the possibility for the algorithm to work with uncertain systems by using interval methods.
- Evaluate the stability of the initial set and find the trajectories which leave the stability region by implementing the SIVIA algorithm (described in Section 2.1.2 and [12]).
- Employ guaranteed integration to calculate the future positions of the trajectories leaving the stability region.
- Find the smallest capture tube which encloses all the trajectories derived from the initial set of values.
- Implement the algorithm and analyse the results for a simple system like a pendulum to illustrate the algorithm capabilities.

The procedure and results presented in this dissertation have been accepted for publishing in the Reliable Computing journal [13] and to SMART 2015 Symposium [14] at The University of Manchester, UK. 16-18 of September 2015.

1.3 Dissertation Outline

This introduction chapter includes the background and motivation for this research, the dissertation aim and the literature review section where the current state of the art in the field is described.

Chapter 2 is a selection of theoretical concepts in order to introduce the concepts of interval analysis, V-Stability, G-Stability and guaranteed integration.

The algorithm and its correspondent theoretical basis are presented in Chapter 3.

Chapter 4 describes the properties of the simple pendulum used as an example of the implementation of the algorithm.

The algorithm is implemented for these systems and the relevant results and discussion are depicted in Chapter 5

Finally, Chapter 6 contains the conclusions of this research and proposed future work.

1.4 Literature Review

Aleksandr Mikhailovich Lyapunov introduced its concept of stability in 1892 [4] but it remained ignored for decades. It was rediscovered by the control engineering community in the late 1950's [3], [15], [16]. In his work, Lyapunov studied the properties of motion and equilibrium and looked for a way to determine the initial values of a problem that make a system state values to remain inside a pre-determined limits. Inspired by Poincaré's work [17], he developed the well-known method to determine if a system is stable laying the foundation of the concept of stability in control theory.

Throughout the years, numerous methods had been developed based on Lyapunov's work [5], [6]. Some of these methods try to extend Lyapunov's theory to linear [18], [19] and nonlinear systems [20]–[22] or adapt it to develop methods for robust [23] and adaptive control [24].

Bolzano, in 1851, was the first to study set theory although his contributions had little influence in the community [25]. Cantor was the true creator of set theory. With his work between 1874 and 1884, he established the basis of this new important field. In 1942 Nagumo proposed its theorem for characterizing positive invariant sets for continuous time systems [26]. It has been used as the base for the consequent developments in that topic [27]–[29].

V-Stability is a method derived from Lyapunov's stability and positive invariant sets and it was introduced by Jaulin and Le Bars in 2012 [9] as a way to determine the stability of time invariant systems in a guaranteed and efficient way and by using interval based algorithms. This method is also capable of dealing with systems which present uncertainty [30].

G-Stability is a new field created as an extension of V-Stability for time dependent systems. This concept was recently introduced by Stancu, Jaulin and Bethencourt [10].

Interval analysis was developed by Moore [31], [32] as a tool to solve the problems caused by the rounding errors produced from the floating point operations in computers. It is a valuable technique to deal with systems with uncertainty [33] since it allows us to perform operations using ranges of values instead of individual values.

Interval analysis is used normally as a substitute of the widely known Monte-Carlo method when guaranteed solutions are needed. Introduced by Ulam and Metropolis in 1949 [34], the Monte-Carlo method is based on the evaluation of a great number of randomly distributed points and because of that it can generate relevant but not guaranteed results [35], [36].

During the last few decades, interval analysis has been expanded to be applied in other fields like robotics [30] and health [37]. One of the most useful tools developed from interval methods [38] is the set inversion via interval analysis (SIVIA) algorithm. It was introduced in 1993 by Jaulin and Walter [12] and it is capable of managing the estimation of nonlinear bounded error [39] without having to assume that it has a certain distribution. One example of the possible application of this technique is the Kalman filter [40], where a Gaussian distribution can be assumed for the system measurements, although is not necessary, in order to produce an approximation of the system real measurements.

Since 2007, Chabert has been developing the Ibex library [41] . It is a C++ library based on interval methods that uses contractor programming to manage and solve non-linear constraints problems. Some of the examples of the use of the library include its application to the study of robotics [42], [43], artificial intelligence [11] and civil structures [44].

The guaranteed integration process is based on solving an initial value problem [45]. One of the possible solutions to this type of problem is to apply the Picard-Lindelöf theorem [46] to obtain a guaranteed solution. This process has been implemented based on the Runge-Kutta method [47] into the Ibex library by Chapoutot with his work in the DynIbex plugin [48]. Guaranteed integration is a more complex and computationally expensive method that

other alternatives such as Euler integration but it limits the propagation of errors and the wrapping effect [49].

It is important to remark that the work presented in this dissertation has been accepted for publication [13] in Reliable Computing, an international, peer reviewed journal dedicated to reliable mathematical computations based on finite representations and guaranteed accuracy [50].

1.5 Notation

Lowercase italic letters x represent real variables. Real vectors $\mathbf{x}$ are written in bold and real matrices in bold and uppercase $\mathbf{X}$. Intervals $[x]$ and interval vectors (boxes) $[\mathbf{x}]$ are represented with brackets. Sets $\mathbb{S}$ are in represented uppercase double symbols. The derivative with respect to time of a function x t is denoted as $\dot{x}$.

Chapter 2

Theoretical Preliminaries

2.1 Introduction to Interval Methods

An interval is a bounded connected subset of $\mathbb{R}$ represented by $[x]$. There is an entire research field based on intervals known as interval analysis [51]. Since an interval is a set, it can be said that interval analysis is a special case of set theory [7].

An interval is normally represented in square brackets and it is defined by its lower ($\underline{x}$) and upper bounds ($\bar{x}$)

$$[x] = [\,\underline{x}\,,\overline{x}\,] \tag{2.1}$$

The lower and upper bounds of an interval can be defined respectively as the largest number on the left of $[x]$ and the smallest number on the right of $[x]$. If an interval shares the same value for its lower and upper bounds it is said to be a punctual interval.

The equivalent of the value 0 in real arithmetic for interval analysis is the empty set $\emptyset$.

Important characteristics of intervals are their width $(w([x]))$ and midpoint $(\mathrm{mid}([\mathrm{x}]))$. These can be calculated as follows:

$$w([x]) \triangleq \overline{x} - \underline{x} \tag{2.2}$$

$$\mathrm{mid}([x]) \triangleq \frac{\underline{x} + \overline{x}}{2} \tag{2.3}$$

Example: Let $[x] = [-1, 3]$, then

$$w([x]) = 3 - (-1) = 4 \tag{2.4}$$

$$\mathrm{mid}([x]) = \frac{-1 + 3}{2} = 1 \tag{2.5}$$

Interval arithmetic includes extended implementations of the four classical operators of real arithmetic. That is addition, subtraction, multiplication and division. As a general rule, for any such binary operator, denoted as $\diamond$, the resultant operation done with the $[x]$ and $[y]$ intervals is equivalent to:

$$[x] \diamond [y] = [\{x \diamond y \in \mathbb{R} \mid x \in [x], y \in [y]\}] \tag{2.6}$$

Example: Let $[x] = [-1, 3]$ and $[y] = [2,5]$, then:

$$[x] + [y] = [-1, 3] + [2, 5] = [1, 8] \tag{2.7}$$

$$[x] - [y] = [-1, 3] - [2, 5] = [-6, 1] \tag{2.8}$$

$$[x] \times [y] = [-1, 3] \times [2, 5] = [-5, 15] \tag{2.9}$$

$$[x]/[y] = [-1, 3]/[2, 5] = [-0.5, 1.5] \tag{2.10}$$

$$[y]/[x] = [2, 5]/[-1, 3] = [-\infty, \infty] \tag{2.11}$$

Let f be a function from $\mathbb{R}$ to $\mathbb{R}$, then $[f]$ is its interval equivalent and can be expressed as

$$[f]([x]) \triangleq [\{f(x) \mid x \in [x]\}] \tag{2.12}$$

For continuous elementary functions such as sin, cos and exp. $[f]([x])$ is equivalent to the image set of $f([x])$.

Example: Let $x = \left[\frac{3\pi}{4}, 2\pi\right]$ and $y = [-3, 4]$, then

$$[f]([x]) = \sin([x]) = \sin\left(\left[\frac{3\pi}{4}, 2\pi\right]\right) = [-1, 0.707] \tag{2.13}$$

$$[f]([y]) = [y]^2 = ([-3, 4])^2 = [0, 16] \tag{2.14}$$

$$[f]([y]) = \sqrt{[y]} = \sqrt{([-3, 4])} = [0, 2] \tag{2.15}$$

A representation of (2.13) is shown in Fig. 1.

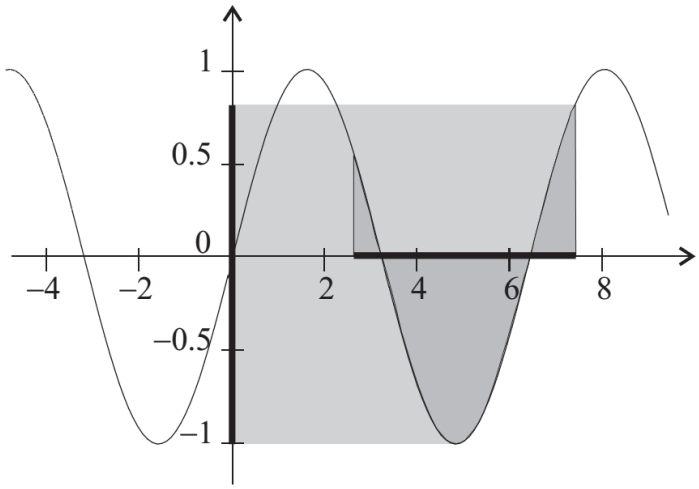

Fig. 1 Solution of $sin([x])$ for $[x] = [3\pi/4\,,\ 2\pi]$. Adapted from [39].

Interval theory is useful to calculate inner and outer approximations of a solution for difficult problems which would be impossible to find by other means [52].

The set of all closed intervals is represented by $\mathbb{IR}$. A vector in $\mathbb{R}^n$ of these intervals is called a box and it is represented by $[\mathbf{x}]$. Using these concepts it is possible to define the interval hull of a subset $\mathbb{A}$ in $\mathbb{R}^n$ as the smallest box which contains $\mathbb{A}$ in $\mathbb{IR}^n$. Another important concept is subpavings. A subpaving is a set of multiple non-overlapping boxes that cover an area of interest $\mathbb{X}$. An example of subpaving for a circle can be seen in Fig. 2

All these notions can also be extended to interval matrices $[\mathbf{X}]$.

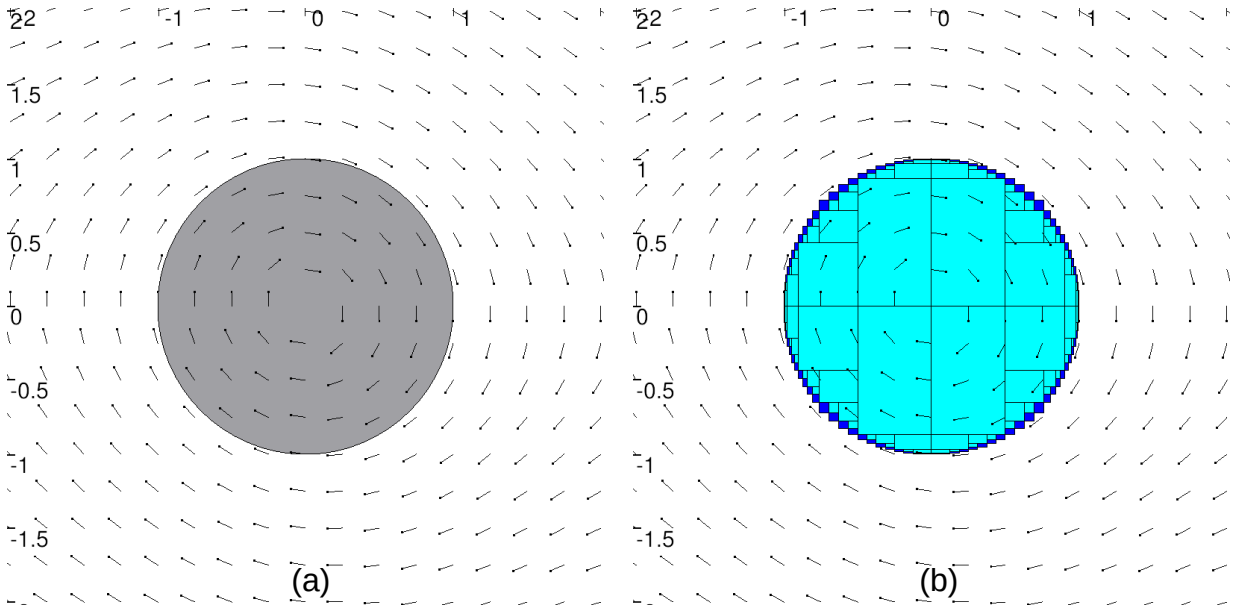

Fig. 2 Representation of the concept of subpaving. Subfigure (a) represents the set defined by a circle of unit radius. Subfigure (b) shows an inner subpaving (clear color), and a boundary subpaving (dark color).

2.1.1 Contractor Programming

A *constrain satisfaction problem* [39] (CSP) $\mathcal{H}$ of n_x interval variables $[x_i] \in \mathbb{R}, i \in \{1, \dots, n_x\}$ related to each other by n_f constraints can be expressed as

$$\mathcal{H}: (f(\mathbf{x}) = 0, \mathbf{x} \in [\mathbf{x}]) \tag{2.16}$$

Where $[\mathbf{x}]$ is an interval vector which includes each $[x_i]$ variable.

Then, its solution set $\mathbb{S}$ is defined as:

$$\mathbb{S} = \{\mathbf{x} \in [\mathbf{x}] \mid f(\mathbf{x}) = 0\} \tag{2.17}$$

A contractor is an operator which replaces $[\mathbf{x}]$ with a smaller counterpart $[\mathbf{x}']$ such that the solution set stays unaffected. All contractors must satisfy the conditions of contractance and completeness. Let $\mathcal{C}$ be a contractor from $\mathbb{IR}^n$ to $\mathbb{IR}^n$, then if it is applied to the box $[\mathbf{x}]$:

$$\mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] \qquad (\text{contractance}) \tag{2.18}$$

$$\mathcal{C}([\mathbf{x}]) \cap \mathbb{S} = [\mathbf{x}] \cap \mathbb{S} \qquad (\text{completeness}) \tag{2.19}$$

An illustration of how a contractor works is shown in Fig. 3.

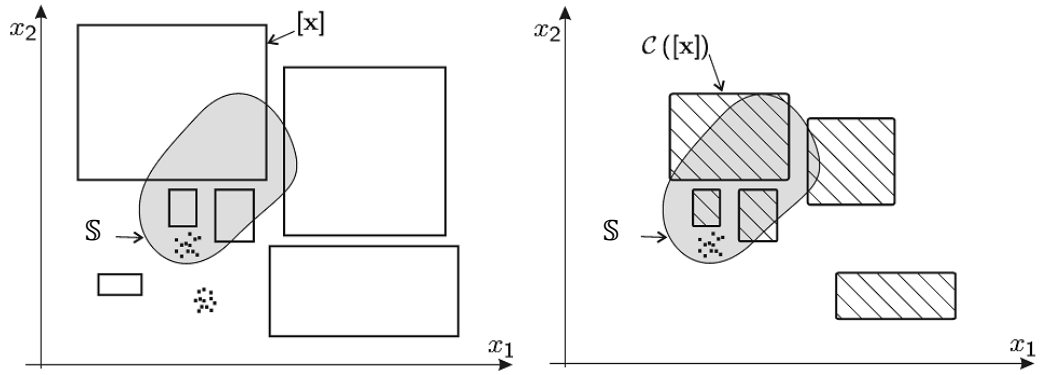

Fig. 3 Subfigure (a) shows the different studied boxes and the solution set before the application of a contractor $\mathcal{C}([\boldsymbol{x}])$. Subfigure (b) shows the resultant contracted boxes. Adapted from [39].

Example: Let $x \in [-\infty, 8], y \in [-\infty, 4], z \in [10, \infty]$ with the constrain $z = x + y$. It is possible to solve this problem by contracting the solution set. To do this, we are going to project the constrain to each variable

$$z = x + y \Longrightarrow z \in [10, \infty] \cap ([-\infty, 8] + [-\infty, 4]) \ldots$$
$$\ldots = [10, \infty] \cap [-\infty, 12] = [10, 12] \tag{2.20}$$

$$x = z - y \Longrightarrow x \in [-\infty, 8] \cap ([10, 12] - [-\infty, 4]) \ldots$$
$$\ldots = [-\infty, 8] \cap [6, \infty] = [6, 8] \tag{2.21}$$

$$y = z - x \Longrightarrow y \in [-\infty, 4] \cap ([10{,}12] - [6{,}8]) \ldots$$
$$\ldots = [-\infty, 4] \cap [2{,}6] = [2{,}4] \tag{2.22}$$

Contractor programming can be used to approximate the solution of a CSP by using set inversion. Let f be a nonlinear function from $\mathbb{R}^n$ to $\mathbb{R}^m$ and $\mathbb{Y}$ a subset of $\mathbb{R}^m$ (e.g a subpaving), then:

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \in \mathbb{Y}\} = f^{-1}(\mathbb{Y}) \tag{2.23}$$

2.1.2 SIVIA Algorithm

Starting from (2.23) and for any $\mathbb{Y}$, it is possible to obtain two subpavings $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$ by using the SIVIA (Set Inverter Via Interval Analysis) algorithm [12].

Let $[\mathbf{x}]$ be the initial search box, $\mathcal{C}_s$ the contractor for the solution set, ϵ the smallest desired size of a box and L the list of solution boxes for the correspondent CSP. The SIVIA algorithm pseudocode is presented in Table 1.

Table 1. SIVIA algorithm for solving a set of non-linear equations

Algorithm: SIVIA $([\mathbf{x}], \mathcal{C}_s, \epsilon, L)$	
1	$[\mathbf{x}] := \mathcal{C}_s([\mathbf{x}])$;
2	if$([\mathbf{x}] = \emptyset)$then return;
3	if$(w([\mathbf{x}]) < \epsilon)$ then
4	$L := L \cup \{[\mathbf{x}]\}$; return;
5	bisect $[\mathbf{x}]$ into $[\mathbf{x}_1]$ and $[\mathbf{x}_2]$
6	SIVIA$([\mathbf{x}_1], \mathcal{C}_s, \epsilon, L)$; SIVIA$([\mathbf{x}_2], \mathcal{C}_s, \epsilon, L)$;

Fig. 4 shows a visual example of the application of the SIVIA algorithm to a circle with a unit radius.

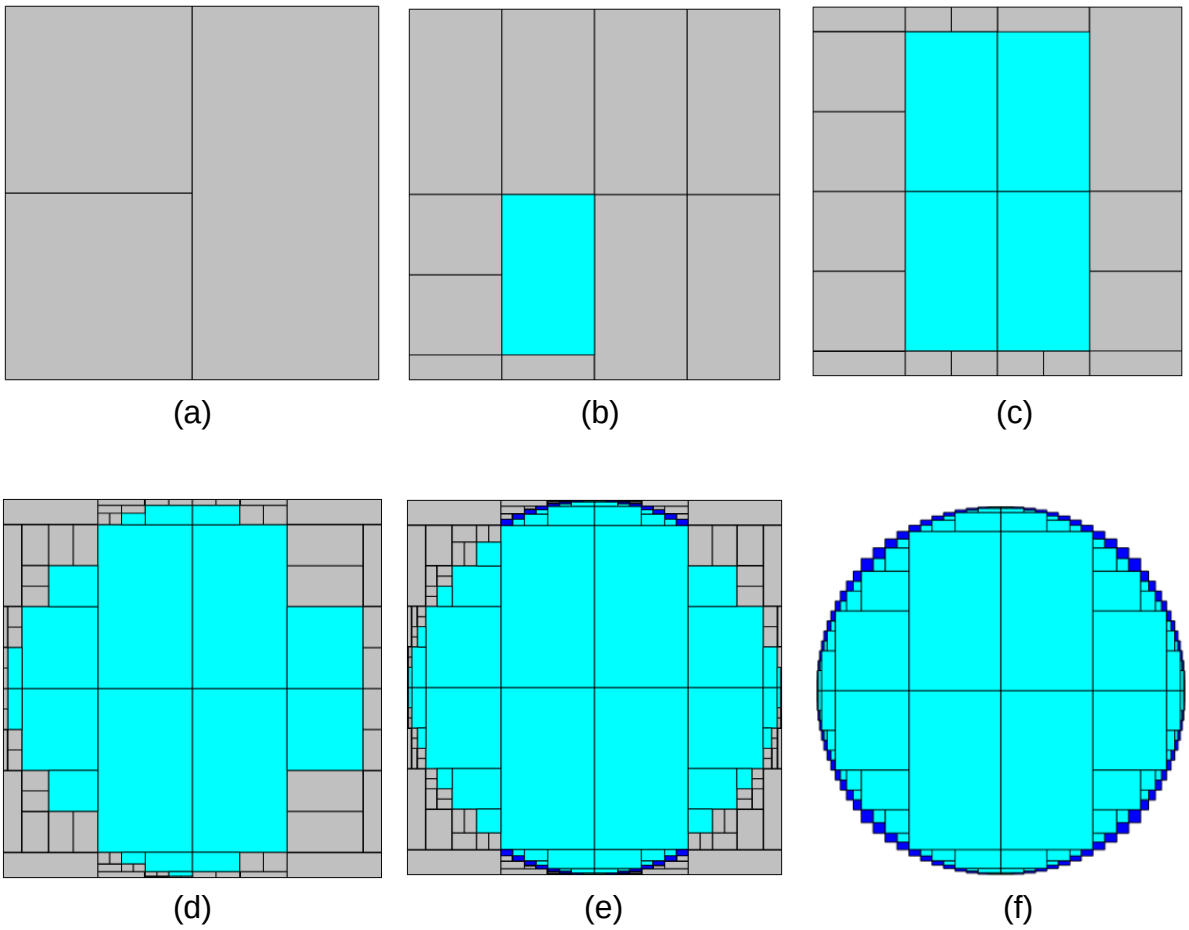

Fig. 4 Representation of the application of SIVIA to a circle. Grey represents the initial box, clear blue is the inner subpaving and dark blue is the boundary subpaving. Subfigures (a) to (e) are ordered chronologically. Subfigure (f) is the final result.

2.2 Lyapunov Stability

A dynamic, autonomous and time invariant system can be described by the following state equation

$$\dot{\mathbf{x}} = f(\mathbf{x}) \tag{2.24}$$

Let D be a neighbourhood of 0 and V_L be a positive definite Lyapunov candidate function. Lyapunov's direct method [6] can establish the stability of a system and can be expressed as a set of constraints:

$$V_L(\mathbf{0}) = 0 \tag{2.25}$$

$$V_L(\mathbf{x}) > 0 \text{ for } D - \{0\} \tag{2.26}$$

$$\dot{V}_L\ (\mathbf{x}) \leq \ \text{for}\ D - \{0\} \tag{2.27}$$

This function $V_L(\mathbf{x})$ is normally derived from the energy of the system and decreases with time. The base idea for Lyapunov stability is that if an equilibrium point of the system is a local minimum for an energy function and the energy of the system is dissipative, then the equilibrium point is (at least locally) stable.

2.3 V-Stability

Analogously to the Lyapunov candidate function, let V be a V-Stability candidate function for our system. The concept of V-Stability [9] is based on Lyapunov stability and it is extended to where V can be negative. The system is said to be V-stable if there exists a $\epsilon > 0$ such that:

$$V(\mathbf{x}) \geq 0 \Rightarrow \dot{V}(\mathbf{x}) \leq \epsilon \tag{2.28}$$

This means that whenever $V(\mathbf{x}) \geq 0$, the system trajectories are attracted, because of the negative time derivative, to the positive invariant set defined by the region which encloses the negative values of $V(\mathbf{x})$. This region is known as a bubble and once a trajectory is inside of it, it will always remain inside. The concept is illustrated in Fig. 5.

A difference between Lyapunov stability and V-Stability is that the Lyapunov candidate function has to be always positive while V-Stability allows the candidate function to have positive and negative values. V-Stability is also weaker than Lyapunov stability in the sense that once a trajectory is inside the negative region of V it can follow any type of trajectory and therefore its time derivative can also take any value.

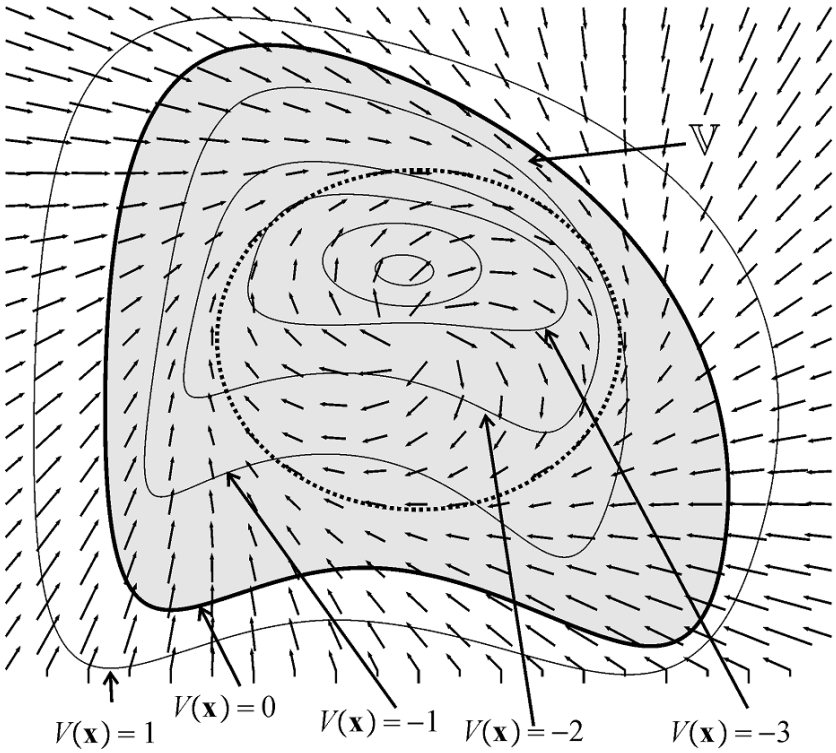

Fig. 5 $V(\boldsymbol{x})$ is the V-Stability candidate function of the system represented by the vector field. The system is said to be V-Stable in the region defined by negative values of $V(\boldsymbol{x})$ (bubble), here represented in grey, since all trajectories from $V(\boldsymbol{x}) \geq 0$ are attracted to it. Notice that there are no trajectories leaving the bubble .Adapted from [9].

Similarly to Lyapunov's stability, V-Stability can be rewritten as a *constrain satisfaction problem* [39]. For a system to be V-stable, the following set of constraints has to be inconsistent, i.e. to have no solutions

$$\begin{cases} \dfrac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot f(\mathbf{x}) \geq 0 \\ V(\mathbf{x}) \geq 0 \end{cases} \tag{2.29}$$

If there are no solutions for the system of inequalities, it means that all the points in the positive region of $V(\mathbf{x})$ have negative derivatives and therefore are attracted to the bubble. Once inside the bubble, they remain there forever since there are no points in its boundary with positive or zero derivative, the vector field on the boundary is only pointing inside the bubble.

Example: Consider the system $\dot{x} = 4 - x$ and $V(x) = x^2 - r^2$. To determine the V-Stability of the system we have to find the values of r that make the system converge into a negative value. This is equivalent to find the values for which the cross-out condition (2.29) is inconsistent.

$$\begin{cases} \dfrac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot f(\mathbf{x}) = 2x \cdot \dot{x} = 2x \cdot (4 - x) \geq 0 \\ V(\mathbf{x}) = x^2 - r^2 \geq 0 \end{cases} \tag{2.30}$$

It can be easily calculated that the system is inconsistent for $r > 4$. Hence, the system will be V-Stable for those r values.

When dealing with systems with uncertainties represented by differential inclusions, the procedure is similar. Let a system with uncertainty be described by the following differential inclusion

$$\dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}) \tag{2.31}$$

Then, the set of constraints which must be inconsistent for the system to be V-stable is

$$\begin{cases} \dfrac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{a} \geq 0 \\ \mathbf{a} \in \mathbf{F}(\mathbf{x}) \\ V(\mathbf{x}) \geq 0 \end{cases} \tag{2.32}$$

2.4 G-Stability

G-Stability is an extension of V-Stability for time variant systems. Consider a nonlinear, time variant system $S_{\mathbf{f}}$ with a target condition $\mathbf{g}(\mathbf{x}, t)$ such that

$$S_{\mathbf{f}} : \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{2.33}$$

$$\mathbf{g}(\mathbf{x}, t) \leq 0 \tag{2.34}$$

G-Stability is based on the concept of capture tubes. A tube $\mathbb{G}$ is a function of time that maps to each time instant a subset of $\mathbb{R}^n$ (bubble). This means that

the target condition creates a time varying bubble that is known as a tube (interval of trajectories). Then, a tube is a capture tube if all the trajectories that enter it remain inside forever

$$\mathbf{x}(t) \in \mathbb{G}(t), \forall \tau > 0 \Rightarrow \mathbf{x}(t+\tau) \in \mathbb{G}(t+\tau) \tag{2.35}$$

A system is G-stable if all the trajectories that enter a capture tube remain inside forever. Consider the system represented in Fig. 6. It can be said that, if we do not consider the dashed trajectory in the bottom of the figure which leaves $\mathbb{G}(t)$ at $(\mathbf{a}, t_a)$, $\mathbb{G}(t)$ is a capture tube since all the trajectories that enter it remain inside. Otherwise, if the dotted trajectory is considered $\mathbb{G}(t)$, is no longer a capture tube.

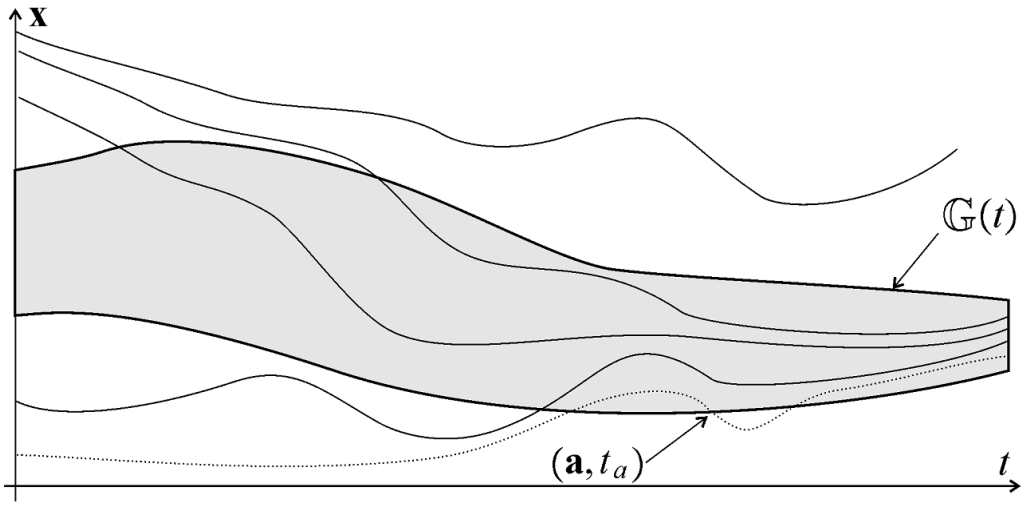

Fig. 6 A tube (represented in grey) and some of the possible trajectories for different initial conditions. If there exist a trajectory which leaves the tube, such as the one represented by the dotted curve, then the tube is not a capture tube.

Analogously to V-Stability, G-Stability can also be expressed as a CSP. The set of constraints (cross-out condition) which must be found to be inconsistent for the system to be G-stable is

$$\begin{cases} \text{(i)} & \dfrac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x},t)\cdot\mathbf{f}(\mathbf{x},t)+\dfrac{\partial g_i}{\partial t}(\mathbf{x},t)\geq 0 \\ \text{(ii)} & gi(\mathbf{x},t)=0 \\ \text{(iii)} & \mathbf{g}(\mathbf{x},t)\leq 0 \end{cases} \tag{2.36}$$

This cross-out condition can also be extended to time variant differential inclusions as shown in (2.32).

$$\begin{cases} \text{(i1)} & \dfrac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x},t)\cdot\mathbf{a}+\dfrac{\partial g_i}{\partial t}(\mathbf{x},t)\geq 0 \\ \text{(i2)} & \mathbf{a}\in\mathbf{F}(\mathbf{x},t)\text{(ii)} \\ \text{(ii)} & gi(\mathbf{x},t)=0 \\ \text{(iii)} & \mathbf{g}(\mathbf{x},t)\leq 0 \end{cases} \tag{2.37}$$

When there are multiple target conditions, the capture tube is defined by their intersection. This way, the cross-out condition can be explained as that there are not any points in the boundaries of this intersection that have positive derivative. Therefore, the trajectories cannot leave the capture tube and remain inside forever. This idea is illustrated in Fig. 7.

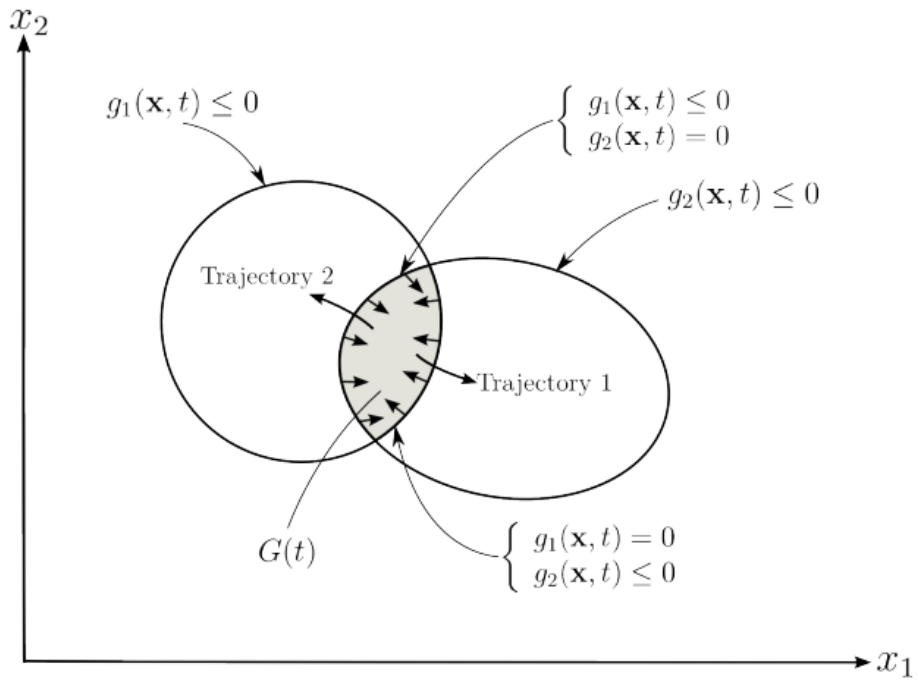

Fig. 7 Cross-out condition for a bi-dimensional capture tube with two constraints. Adapted from [10]

It is possible to use the concept of G-Stability to determine the safety of a squad of robots by checking the safety of their respective capture tubes [10]. Let S^a and S^b be two nonlinear time variant systems with $\mathbb{G}^a$ and $\mathbb{G}^b$ as their respective capture tubes, then for all $\mathbf{x}^a$ the safety problem can be expressed as a CSP with the following constraints

$$g^a(\mathbf{x}^a, t) \leq 0 \tag{2.38}$$

$$g^b(\mathbf{x}^a, t) < 0 \tag{2.39}$$

This constrain system must be inconsistent for the systems to be safe. In other words, if the tubes do not share any points and they do not collide therefore neither do the systems.

2.5 Guaranteed Integration

Due to its properties, a tube can be very difficult to represent in a computer. Because of this reason, for numerical calculations, the tube is represented by inner and outer approximations [39], [53].

Tubes can also be represented by interval polynomials [31], this presents some advantages such that all the operations that are possible with scalar polynomials are also possible with them, with the only exception of the derivative.

Example: Let $\mathbb{G}(t) = \sin(\omega t) \cdot t \in [-1,1] \cdot t$

It is clear that we cannot conclude that

$$\frac{1}{\omega}\cos(\omega t) \in [-1,1] \tag{2.40}$$

In order to implement the algorithm developed in this research, a technique to integrate all the possible solutions of a state equation or a differential inclusion is needed. Guaranteed integration [46], [53] achieves this objective.

The guaranteed integration method is developed around the Brouwer theorem.

Brouwer theorem: any continuous function f(x) mapping a compact convex set $\mathbb{X}$ to itself has a fixed point

$$\exists x \in \mathbb{X} \mid f(x) = x \tag{2.41}$$

Example: Consider $f(x) = \sin(x) \cdot \cos(x)$ and $\mathbb{X} = [-2,2]$. Since,

$$f([-2,2]) \subset \sin([-2,2]) \cdot \cos([-2,2]) = [-1,1] \subset \mathbb{X} \tag{2.42}$$

Then, applying the Brouwer theorem, we have

$$\exists x \in [-2,2] \mid \sin(x) \cdot \cos(x) = x \tag{2.43}$$

It is possible to extend it to the differential inclusion case [54], [55] by using its parametric version.

Parametric Brouwer theorem: if $f: \mathbb{X} \times \mathbb{U} \to \mathbb{X}$, where $\mathbb{X}$ is a convex compact set and f is continuous with respect to $x \in \mathbb{X}$, then

$$\forall u \in \mathbb{U}, \exists x \in \mathbb{X} \mid f(x,u) = x \tag{2.43}$$

Consider a nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$ be a Lipschitz continuous function and its known initial condition $\mathbf{x}_0^*$. The Picard-Lindelöf operator [45] is able to calculate the interval enclosure for the trajectory $\mathbf{x}^*(t)$ and it is defined as

$$\mathcal{T}: \mathbf{x}(t) \to \left(t \mapsto \mathbf{x}_0^* + \int_0^t \mathbf{f}\big(\mathbf{x}(\tau)\big) d\tau \right) \tag{2.44}$$

Because $\mathbf{f}$ is Lipschitz continuous, the Picard-Lindelöf operator has a unique fixed point correspondent to the solution $\mathbf{x}^*(t)$. For an interval tube $[\mathbf{x}](t)$, each $\forall t, [\mathbf{x}](t)$ is a box of $\mathbb{R}^n$. From the Brouwer theorem

$$\mathcal{T}\big([\mathbf{x}](t)\big) \subset [\mathbf{x}](t) \Rightarrow \mathbf{x}^*(t) \in [\mathbf{x}](t) \tag{2.45}$$

Fig. 8 provides a representation of the tube $[\mathbf{x}](t)$ and its correspondent applied Picard-Lindelöf operator $\mathcal{T}\big([\mathbf{x}](t)\big)$. Notice that, because of the form of $\mathcal{T}$, at the initial instant the resultant tube $\mathcal{T}\big([\mathbf{x}](t)\big)$ is very thin. Notice that, in this case, $\mathcal{T}\big([\mathbf{x}](t)\big) \subset [\mathbf{x}](t)$ is only valid for $t \in [0, t_1]$. If we apply this restriction to $\mathcal{T}$, we get the inclusion. Hence

$$\forall t \in [0, t_1], \mathbf{x}^*(t) \in \mathcal{T}([\mathbf{x}](t)) \tag{2.46}$$

Where t_1 can be defined as

$$t_1 = \sup\big\{ t \in \mathbb{R}^+ \mid \forall \tau \in [0,t], \mathcal{T}\big([\mathbf{x}](t)\big) \subset [\mathbf{x}](t) \big\} \tag{2.47}$$

Obviously, the operator can be called several times

$$\forall i \geq 0, \forall t \in [0, t_1], \mathbf{x}^*(t) \in \mathcal{T}^i([\mathbf{x}](t)) \tag{2.48}$$

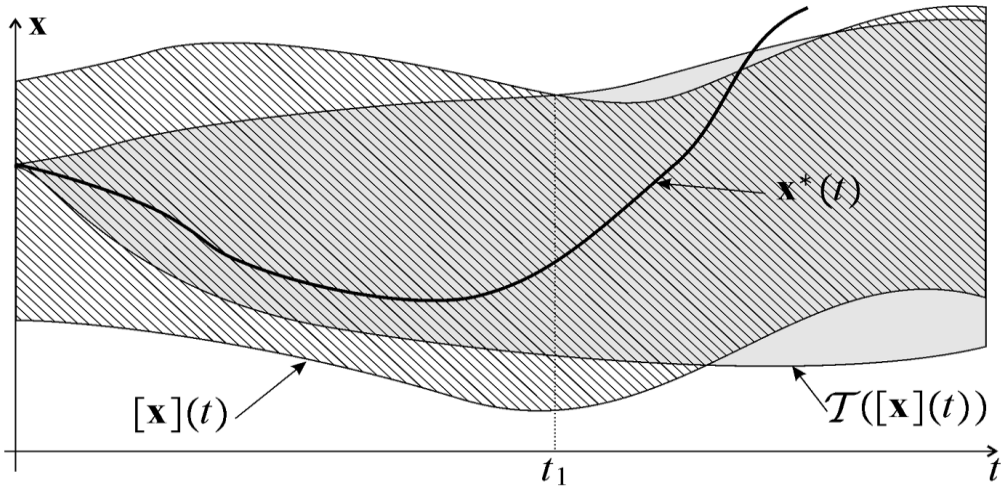

Fig. 8 Illustration of the Picard-Lindelöf operator for the tube $[\boldsymbol{x}](t)$

Guaranteed integration can be extended to deal with a system with uncertainties [54], [55]. Let $\mathbf{x}_0$ be the uncertain system initial state and that the system has an uncertain input vector $\mathbf{u}(t)$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.49}$$

For this system $\mathbf{x}_0 \in [\mathbf{x}_0]$ and $\mathbf{u}(t) \in [\mathbf{u}](t)$. It is possible to set

$$\mathbf{F}(\mathbf{x}, t) = \{\mathbf{f}(\mathbf{x}, \mathbf{u}) \mid \mathbf{u}(t) \in [\mathbf{u}](t)\} \tag{2.50}$$

Assuming that $\mathbf{f}$ if Lipschitz, the Picard-Lindelöf operator for differential inclusions can be defined as

$$\mathcal{T}_{\mathbf{x}_0,\mathbf{u}}: \mathbf{x}(t) \to \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau))\, d\tau \tag{2.51}$$

Guaranteed integration is the method chosen for our algorithm because of its guaranteed properties. It is more computationally expensive than other methods such as Euler integration but, due to its more advanced consideration of the system, it is able to present smaller errors derived from wrapping effects [49].

2.5.1 Wrapping effect

Wrapping effect is the result of the integration of errors introduced by the different steps in the simulation of the future states of a system.

A simpler alternative to guaranteed integration is Euler integration, for a system $[\dot{\mathbf{x}}] = f([\dot{\mathbf{x}}])$, it can be implemented by transforming the system from continuous time to discrete time such that

$$[\dot{\mathbf{x}}] = \frac{[\mathbf{x}]_{k+1} - [\mathbf{x}]_k}{dt} \rightarrow [\mathbf{x}]_{k+1} = [\mathbf{x}]_k + dt \cdot [\dot{\mathbf{x}}] \tag{2.52}$$

Example: Consider the following system

$$[\dot{\mathbf{x}}] = \begin{pmatrix} [x_2] \\ -[x_1] \end{pmatrix} \tag{2.53}$$

Euler integration can be used to obtain the future states of the system

$$[\mathbf{x}]_{k+1} = [\mathbf{x}]_k + dt \begin{pmatrix} [x_2] \\ -[x_1] \end{pmatrix} \tag{2.54}$$

This method generates the results shown in Fig. 9. It can be seen how the initial box is transformed with each iteration. Because of this, it is needed to enclose the result in a new box which contains extra values. This example produces a large wrapping effect that renders it unusable for our algorithm.

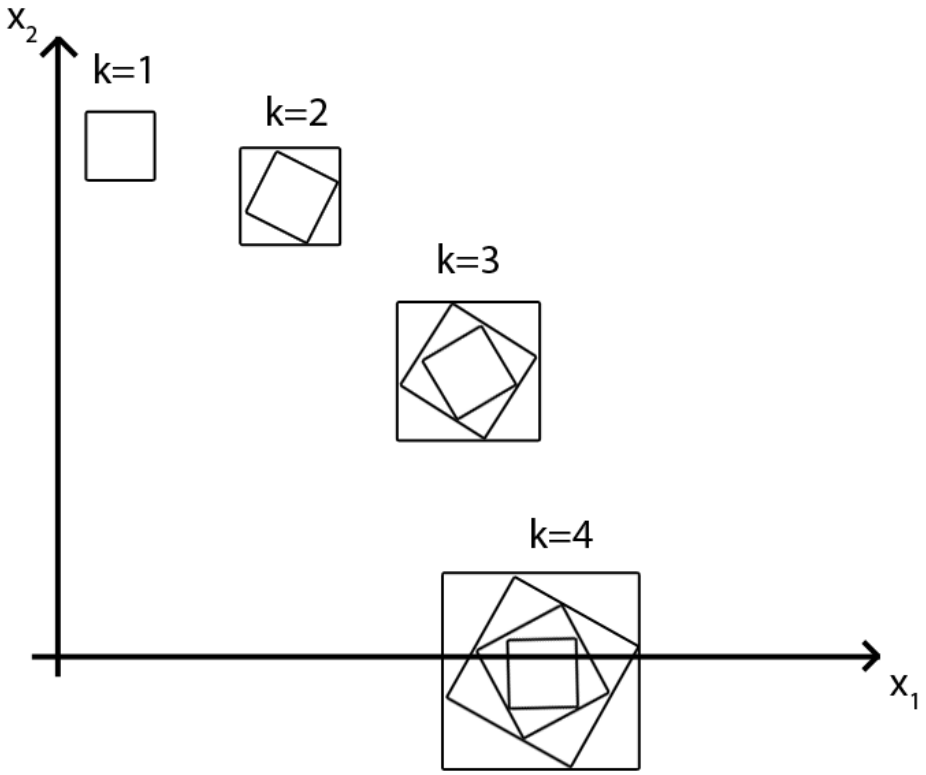

Fig. 9 Representation of the wrapping effect derived from the use of Euler integration.

Chapter 3

Computing Capture Tubes

It is simple to prove that a candidate tube $\mathbb{G}$ is a capture tube by checking the cross-out condition, i.e., by proving the inconsistency of a set of inequalities. Interval analysis is a powerful tool which can be used to this. However, for complex systems, it is difficult to find a candidate tube that is a capture tube. The main contribution of this research is to provide a method to calculate capture tubes. The algorithm starts from a non-capture tube $\mathbb{G}$ and then finds the smallest capture tube $\mathbb{G}^+$ that encloses it. This is achieved by creating a guaranteed envelope around the initial tube $\mathbb{G}$ that also encloses all the future states of the trajectories that leave the initial tube.

3.1 Lattice and Capture Tubes

A lattice [7] is a partially ordered set in which every pair of components have a unique supremum and infimum. These two concepts can also be referred as upper and lower bounds respectively.

Consider $\mathbf{a}(t)$ and $\mathbf{b}(t)$ as two trajectories. It is possible to define a tube as an interval of trajectories since the set of trajectories from $\mathbb{R}$ to $\mathbb{R}^n$ is a lattice following the relation

$$\mathbf{a}(t) \leq \mathbf{b}(t) \Leftrightarrow \forall t, \forall i, a_i(t) \leq b_i(t) \qquad (3.2)$$

It is also possible to affirm that a set of tubes $(\mathbb{T}, \subset)$ is a lattice since the set of subsets of $\mathbb{R}^n$ is also a lattice with respect to the subset inclusion $\subset$.

Theorem: Consider a state space system $S_{\mathbf{f}}: \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ or a differential inclusion $S_{\mathbf{F}}: \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, t)$. The set of capture tubes $(\mathbb{T}_c, \subset)$ for $S_{\mathbf{f}}$ or $S_{\mathbf{F}}$ is a sublattice of the set of tubes $(\mathbb{T}, \subset)$.

Proof: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two capture tubes of a nonlinear dynamic system or a differential inclusion. If there exists a trajectory $\mathbf{x}(t)$ that belongs to both $\mathbb{G}_1$ and $\mathbb{G}_2$, then $\mathbf{x}(t)$ will never leave either one of them. Therefore, the intersection of $\mathbb{G}_1$ and $\mathbb{G}_2$ is also a capture tube. Using the same logic is also possible to do affirm that the union of the two capture tubes is also a capture tube. Hence, we can reason that $(\mathbb{T}_c, \subset)$ is a sublattice of $(\mathbb{T}, \subset)$ since $\mathbb{G}_1 \cap \mathbb{G}_2$ is the largest capture tube included in both $\mathbb{G}_1$ and $\mathbb{G}_2$ and $\mathbb{G}_1 \cup \mathbb{G}_2$ is the smallest capture tube that encloses both. ■

We can define a new operator that will correspond to the smallest capture tube which encloses $\mathbb{G}$

$$\mathrm{capt}\big(\mathbb{G}(t)\big) = \cap \left\{ \overline{G}(t) \in \mathbb{T}_c \;\middle|\; \mathbb{G}(t) \in \overline{\mathbb{G}}(t) \right\} \tag{3.1}$$

Therefore the set of tubes is also a lattice with respect to the inclusion $\subset$ and we can define intervals of tubes. This concept is key in the development of this research and allows us to characterize the tubes that we want to calculate using inner and outer approximations. Otherwise they may not be representable in a computer and very difficult to use in calculations.

3.2 Computing Capture Tubes

An interval of tubes $[\mathbb{G}]$ is a subset of the set of tubes $\mathbb{T}$ which satisfies

$$[\mathbb{G}] = \{\mathbb{G} \in \mathbb{T} \mid \mathbb{G} \subset [\mathbb{G}]^+ \text{ and } \mathbb{G} \supset [\mathbb{G}]^- \tag{3.3}$$

Where $[\mathbb{G}]^+$ represents the smallest outer bound of $[\mathbb{G}]$ and $[\mathbb{G}]^-$ the largest inner bound of $[\mathbb{G}]$. Then, $\mathbb{IT}$ represents the set of intervals of tubes.

The main problem that needs to be solved is that given an initial tube $\mathbb{G}(t)$ we need to calculate the interval of tubes $[\mathbb{G}^-(t), \mathbb{G}^+(t)] \in \mathbb{IT}$ such that

$$\mathrm{capt}\big(\mathbb{G}(t)\big) \in [\mathbb{G}^-(t), \mathbb{G}^+(t)] \tag{3.4}$$

The main difficulty is to get a $\mathbb{G}^+(t)$ that is not too large since we can consider $\mathbb{G}^-(t) = \mathbb{G}(t)$ due to the fact that $\mathbb{G}(t) \subset \mathrm{capt}(\mathbb{G}(t))$. This idea is illustrated in Fig. 10.

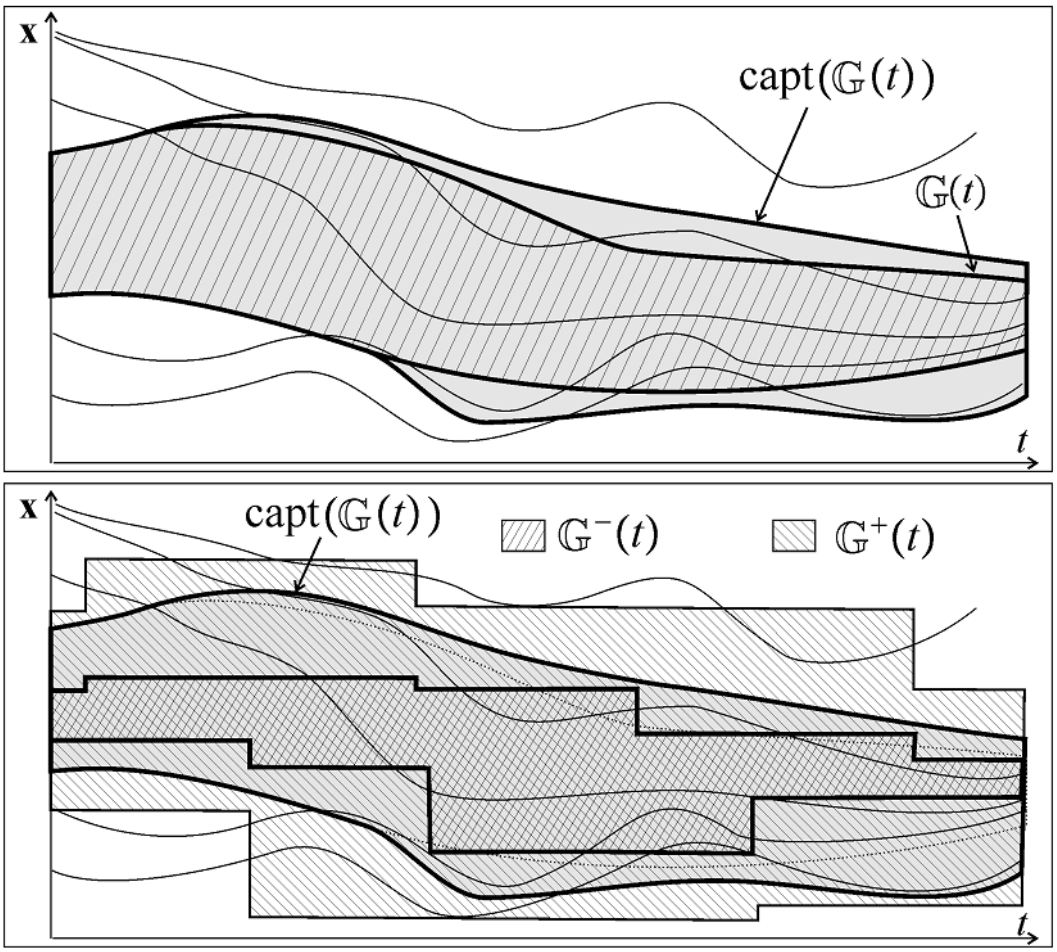

Fig. 10 Our solution capture tube $capt(\mathbb{G}(t))$ is enclosed between its inner $\mathbb{G}^-$ and outer $\mathbb{G}^+$ approximations. These approximations form an interval of tubes $[\mathbb{G}^-, \mathbb{G}^+]$.

The concept of the *flow of a system* $\mathcal{S}_{\mathbf{f}}: \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ can be defined as a function $\phi_{t_0,t_1}: \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \Rightarrow \phi_{t_0,t_1}\big(\mathbf{x}(t_0)\big) = \mathbf{x}(t_1) \tag{3.5}$$

The meaning of this expression is that if the system $\mathcal{S}_{\mathbf{f}}$ has a solution $\mathbf{x}(t)$, it is possible for it 'to move' following the flow from the state at t_0 to the state at t_1.

Similarly, for a differential time inclusion $\mathcal{S}_{\mathbf{F}}: \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, t)$, the flow is a function $\phi_{t_0,t_1}: \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$, here $\mathcal{P}(\mathbb{R}^n)$ represents the set of subsets of $\mathbb{R}^n$ such that

$$\dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, t) \Leftrightarrow \mathbf{x}(t_1) \in \phi_{t_0,t_1}\big(\mathbf{x}(t_0)\big) \tag{3.6}$$

This means that using the flow we can find all the system states that can be reached from the initial state using trajectories that belong to the differential inclusion $\mathcal{S}_{\mathbf{F}}$.

Theorem: Consider the system $\mathcal{S}_{\mathbf{f}}: \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ and its associated flow function $\phi_{t_0,t}$.The tube defined by

$$\begin{aligned} \mathbb{C}(t): t \ \to \ \{\mathbf{x} \mid \exists(\mathbf{x}_0, t_0), \mathbf{x}_0 \in \mathbb{G}(t_0), t \ \geq \ t_0, \ldots \\ \ldots \ \mathbf{x} = \phi_{t_0,t}\ (\mathbf{x}_0)\} \end{aligned} \tag{3.7}$$

corresponds to $\mathrm{capt}(\mathbb{G}(t))$.

Proof: To show that $\mathbb{C}(t)$ is the smallest capture tube which encloses our initial guess $\mathbb{G}(t)$, it is must be proven that $\mathbb{C}(t)$ contains $\mathbb{G}(t)$, that $\mathbb{C}(t)$ is a capture tube and that $\mathbb{G}^+(t)$ is the smallest one.

In order to prove that $\mathbb{G}(t)$ is a subset of $\mathbb{C}(t)$ it is enough to take $\mathbf{x}_0 = \mathbf{x}$ and $t_0 = t$.

Next, to check if $\mathbb{C}(t)$ is a capture tube it is possible to use the flow concept to check that for any $x_t \in \mathbb{C}(t)$ we have the following expression

$$\exists(\mathbf{x}_0, t_0), \mathbf{x}_0 \in \mathbb{G}(t_0), t \geq t_0, \mathbf{x}_t = \phi_{t_0,t}\ (\mathbf{x}_0) \tag{3.8}$$

Consider $\tau > 0$ and define $x_{t+\tau} = \phi_{t,t+\tau}(\mathbf{x}_t)$. Such that, the expression above is transformed to

$$\exists\ (\mathbf{x}_0, t_0), \mathbf{x}_0 \ \in \mathbb{G}(t_0), t \ \geq \ t_0, \mathbf{x}_{t+\tau} = \phi_{t_0,t+\tau}(\mathbf{x}_0) \tag{3.9}$$

Therefore, it has been proven that

$$\mathbf{x}_t \in \mathbb{C}(t), \tau \geq 0 \Rightarrow \phi_{t,t+\tau}(\mathbf{x}_t) \in \mathbb{C}(t+\tau) \tag{3.10}$$

And therefore $\mathbb{C}(t)$ is a capture tube.

For the last condition, that is for $\mathbb{C}(t)$ to be the smallest capture tube, it is possible to do the proof by contradiction. Consider $\overline{\mathbb{G}}(t)$ as a capture tube such that $\overline{\mathbb{G}}(t) \supset \mathbb{G}(t)$ and which is strictly enclosed in $\mathbb{C}(t)$. This means that $\exists(t_1, \mathbf{x}_1), \mathbf{x}_1 \in \mathbb{C}(t_1)$ and $\mathbf{x}_1 \notin \overline{\mathbb{G}}(t_1)$. Then, from (3.7), it can be seen how there is a trajectory that leaves the tube $\overline{\mathbb{G}}(t)$. Hence, it is proven that $\overline{\mathbb{G}}(t)$ is not a capture tube and $\mathbb{C}(t)$ is the smallest capture tube which encloses $\mathbb{G}(t)$. ■

This reasoning can also be applied to differential inclusions $\mathcal{S}_{\mathbf{F}}: \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, t)$ by considering the tube defined by

$$\begin{aligned} \mathbb{C}(t): t \rightarrow \{\mathbf{x} \mid \exists(\mathbf{x}_0, t_0), \mathbf{x}_0 \in \mathbb{G}(t_0), t \geq t_0, \ldots \\ \ldots \mathbf{x} \in \phi_{t_0,t}(\mathbf{x}_0)\} \end{aligned} \tag{3.11}$$

Theorem: Consider the system $\mathcal{S}_{\mathbf{f}}: \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$. Then

$$\text{capt}\big(\mathbb{G}(t)\big) = \mathbb{G}(t) \cup \Delta\mathbb{G}(t) \tag{3.12}$$

where

$$\begin{aligned} \Delta\mathbb{G}(t) = t \mapsto \{\mathbf{x} \mid \exists(\mathbf{x}_0, t_0) \text{ satifying } (2.22), t \geq t_0, \ldots \\ \ldots x = \phi_{t_0,t}(\mathbf{x}_0) \; and \; \mathbf{x} \notin \mathbb{G}(t)\} \end{aligned} \tag{3.13}$$

Proof: In order to calculate $\Delta\mathbb{G}(t)$ we need to find all the pairs $(\boldsymbol{x}, t)$ outside the initial tube $\mathbb{G}(t)$ that can be reached from a pair $(\boldsymbol{x}_a, t_a)$ that belongs to $\mathbb{G}(t)$. The corresponding trajectory will cross the boundary of the candidate tube $\mathbb{G}(t)$ at instant t_0 at the state x_0, i.e., (x_0, t_0) satisfies (2.36). This is illustrated in Fig. 11. ■

This can also be adapted for differential inclusions.

Theorem: Consider the system $\mathcal{S}_{\mathbf{F}}: \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, t)$. Then

$$\mathrm{capt}\big(\mathbb{G}(t)\big) = \mathbb{G}(t) \cup \Delta\mathbb{G}(t) \tag{3.12}$$

where

$$\begin{aligned}\Delta\mathbb{G}(t) = t \mapsto \{\mathbf{x} \mid \exists(\mathbf{x}_0, t_0) \text{ satifying } (2.22), t \geq t_0, \ldots \\ \ldots\ x \in \phi_{t_0,t}(\mathbf{x}_0)\ and\ \mathbf{x} \notin \mathbb{G}(t)\}\end{aligned} \tag{3.13}$$

Proof: The proof is a direct consequence of Theorem (3.12). ■

Consequences: As illustrated in Fig. 11, the interval $[\mathbb{G}^-(t), \mathbb{G}^+(t)]$ that approximates $\mathrm{capt}(\mathbb{G}(t))$ is defined by the tube $\mathbb{G}^-(t) = \mathbb{G}(t)$ and its union with the enclosure $\Delta\mathbb{G}(t)$ of all the trajectories generated from $(\mathbf{x}_0, t_0)$ that satisfy the cross-out condition.

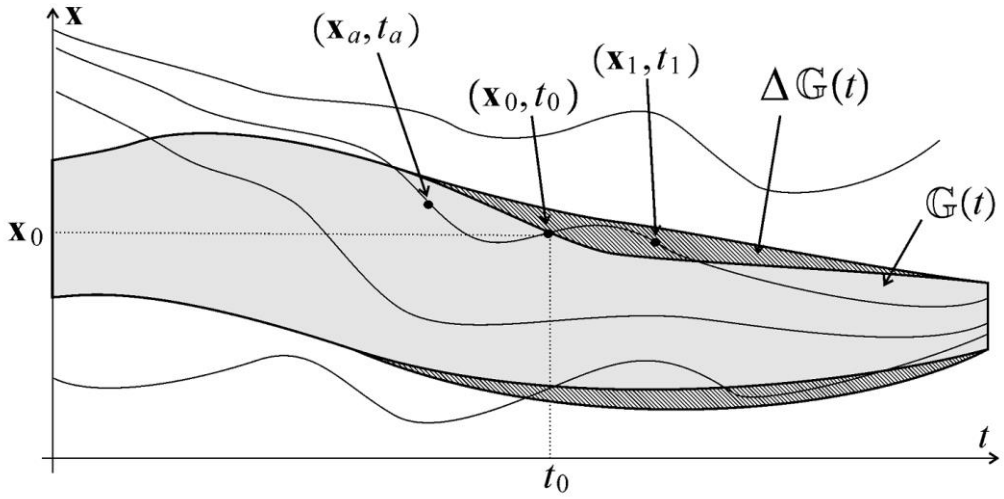

Fig. 11 All the trajectories which leave the tube are enclosed by $\Delta\mathbb{G}(t)$.

3.3 Implementation

The algorithm has been built in C++ using the QT 4.8.6 framework [56], [57]. This framework was chosen due to its advanced handling capabilities of files

[58] and data structures [59]. The source code of the algorithm is publicly available at

https://github.com/dlopezmadrid/tubibex

The development was done in a 64 bits Linux Ubuntu 14.04 environment [60]. The software packages used in this project are ibex-lib 2.1.16 [41] with SoPlex 1.7.2 [61] as the solver and DynIbex for ibex-lib 2.1.16 [48] for the guaranteed integration. In order to represent the solutions, we have chosen to use VIBes 0.2.0 [62] because of its easy implementation in C++.

Notice that the algorithm has been parametrically implemented and therefore can handle systems of any number of dimensions. The only constrain for more complex system is the required extra computation time.

This implementation of the algorithm has around 1400 lines of code and can be divided in initialization of the program, parameter configuration, SIVIA, guaranteed integration and display of results. Each one of these parts is described below.

3.3.1 Initialization

During this section, the algorithm checks if the text files containing the $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{g}(\mathbf{x}, t)$ functions exist. If they do not exist, the program creates the files with default the functions. These functions are written using the minibex syntax [63] for an easy import to the algorithm. If the function files exist, the program checks if there are any syntax error and, if they are correct, loads them to the memory, otherwise displays an error. This section also initialises the VIBes viewer for its later use.

3.3.2 Parameter Configuration

The user can configure some of the program values such as the initial search box, minimum interval width for each dimension, system time, time step, how far in the future the boxes are simulated and set a limit for the number of processed boxes.

Then, there are also visualization options such as show the internal subpaving of the $\mathbf{g}(\mathbf{x}, t)$ function, mesh precision for the vector field, represent the results for future time from the simulation, draw the boxes in real time as they are being processed, add a delay between real time draws to be able to see the process better, set the plane to represent (e.g. XY, XZ, YZ) and export the figures to an image file.

After the load of these user defined parameters, the program performs some automatic configuration based on the user choices and the functions loaded from the text files. One of these tasks is to create the initial box for the SIVIA algorithm. This box is an interval vector which contains the initial interval values for each one of the system dimensions and the time.

3.3.3 SIVIA Algorithm

This section starts by initializing the required components to perform the SIVIA algorithm described in 2.1.2. It uses the provided $\mathbf{g}(\mathbf{x}, t)$ function to create the constraints that define the cross-out condition and puts the initial box in the queue of boxes to be processed. Then, it defines the contractors which will be used in the SIVIA algorithm.

Next, it performs a recursive SIVIA algorithm that contracts and bisects the boxes until they cannot be contracted more or their widths are smaller than the user defined ϵ for each dimension.

This process divides the boxes in safe and unsafe boxes based on the result of the cross-out condition constraints. A box is unsafe it is has any solution for the cross-out condition, otherwise, the box is said to be safe, even if it is outside the $\mathbf{g}(\mathbf{x}, t) < 0$ region.

3.3.4 Integration

The integration process is done using DynIbex. For each unsafe box, the algorithm creates an initial value problem [46], [47] which is solved, using a DynIbex simulation, for the time horizon defined by the user. This process obtains all the possible system solutions for the initial values defined by the

correspondent unsafe box. The list of solutions is stored in memory in order to represent them later.

3.3.5 Results Visualization

After all the unsafe boxes have been processed, the system prints result statistics like the total number of boxes, the number of unsafe boxes and the elapsed time.

Then, it uses VIBes to draw the results on the screen. With our implemented function, VIBes can handle different colours, plot different planes and represent the vector field. A series of figures are drawn and exported as defined by the user.

Chapter 4

Case Study

The pendulum, as illustrated in Fig. 12, can be described by the following state equations

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\sin(x_1) - v \cdot x_2 \end{cases} \tag{4.1}$$

Where x_1 represents the angle with respect to the vertical axis, x_2 the correspondent angular speed, and v is a fixed damping factor. For this example we will consider $v = 0.15$.

In order to find a capture tube (positive invariant tube) for such a system, the classical approach is to study its energy levels. For the pendulum

$$E(\mathbf{x}) = \frac{1}{2}\dot{x}_1^2 - \cos(x_1) + 1 = \frac{1}{2}x_2^2 - \cos(x_1) + 1 \tag{4.2}$$

Because of the damping factor, this energy will decrease with time and therefore it can be a good candidate function $\mathbf{g}(\mathbf{x}, t)$. We can define our candidate tube as

$$\mathbf{g}(\mathbf{x}) = E(\mathbf{x}) - 1 = \frac{1}{2}x_2^2 - \cos(x_1) \tag{4.3}$$

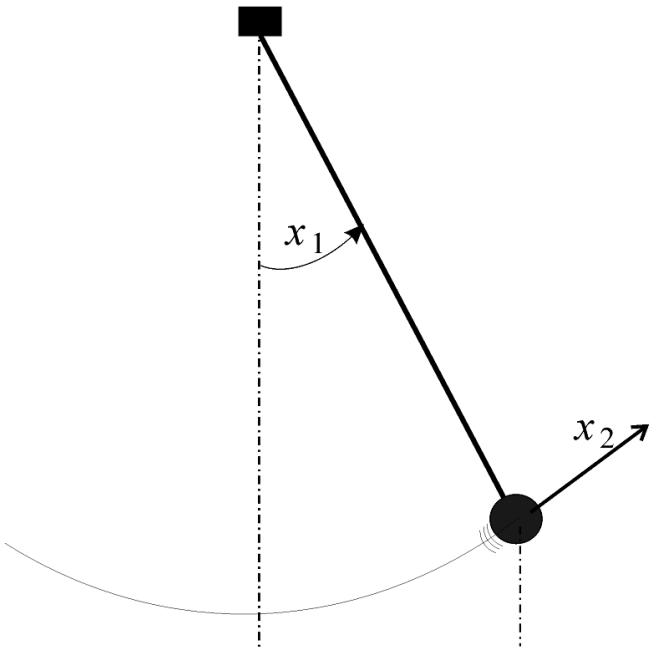

Fig. 12 Representation of the pendulum system. x_1 is the pendulum angle and x_2 its angular velocity.

Now it is possible to evaluate the cross-out condition for this candidate tube

$$\begin{cases} (\sin(x_1) \quad x_2) \begin{pmatrix} x_2 \\ -\sin(x_1) - 0.15 \cdot x_2 \end{pmatrix} = -0.15 \cdot x_2^2 \geq 0 \\ \frac{1}{2}x_2^2 - \cos(x_1) = 0 \end{cases} \tag{4.4}$$

Note that, in this case, $\mathbf{g}(\mathbf{x})$ is scalar and therefore (2.36) (ii) and (2.36) iii are equivalent. The cross-out condition has two solutions $\left(\mathbf{x} = \left(\pm\frac{\pi}{2}, 0\right)\right)$ and therefore the chosen candidate is not a capture tube. Notice that even for this simple time-invariant systems with only two dimensions for which the classical approach has a good intuition of a function (based on the systems energy) which almost always decreases, it is difficult to get a capture tube.

To solve this problem it is possible to apply the algorithm proposed in this dissertation. It does not depend on the energy of the system, which can only be applied to a small class of systems. Instead, we can choose an initial candidate tube the one associated with the function

$$\mathbf{g}(\mathbf{x}) = \frac{x_1^2}{1.75^2} + \frac{x_2^2}{0.75^2} - 1 \tag{4.5}$$

This function corresponds to an ellipse centred at $(0,0)$ which crosses the x_1 axis at $\pm\ 1.75$ and the x_2 axis at ± 0.75. The only requirement for this function is that it is continuous and derivable.

It is important to remark that we have chosen a time-invariant tube in order to be able to represent the results in static images. In this case both $\mathbb{G}$ and $\Delta\mathbb{G}$ are time-invariant and both of them become subsets of $\mathbb{R}^2$.

Chapter 5

Results and Discussion

Results from the pendulum example can be seen in Fig. 13. Subfigure (a) represents our initial tube (4.5). Then subfigure (b) shows the inner and boundary subpavings for the initial tube and therefore also for the inner approximation of $\mathrm{capt}(\mathbb{G})$. Next, subfigure (c) represents the boxes that enclose the points which satisfy the cross-out conditions. Subfigure (d) shows all the previous subpavings together. In subfigure (e), it can be seen the guaranteed integration of the trajectories that leave the initial tube. Finally, subfigure (f) shows a superposition of $\Delta\mathbb{G}$ and $\mathbb{G}$.The union of these boxes constitutes the outer approximation $\mathbb{G}^+$ of the capture tube $\mathrm{capt}(\mathbb{G})$ and therefore, the real capture tube belongs to the interval defined by $[\mathbb{G}^-, \mathbb{G}^+]$.

For the generation of Fig. 13, the system used is the one presented in (4.1) with the target condition described in (4.5). The initial box was set to $[\mathbf{x}]_0 = [-\infty, \infty], [-\infty, \infty]$ and the minimum box size was set to 0.1. The algorithm produced a total of 418 boxes to find the inner (124 boxes) and boundary (128 boxes) subpavings of $\mathbf{g}(\mathbf{x})$ and the boxes that satisfy the cross-out conditions (48 boxes). Then, each of these unsafe boxes, was integrated using guaranteed integration for a time horizon of three seconds in the future. The algorithm took 3 minutes and 21 seconds to complete all the calculations.

Fig. 14 shows a comparison of the wrapping effect derived from the use of guaranteed integration (subfigure (a)) and Euler integration (subfigure (b)). This shows that guaranteed integration is able to limit the wrapping effect and produce more accurate results and that Euler integration is not suitable for this application.

This algorithm allows us to guarantee the safety of the system if it starts in the studied initial conditions. This can be done because we have calculated all the possible system trajectories that can be reached from all the initial states. This kind of guaranteed result is impossible to obtain with methods that rely on studying a large number of individual points like the Monte-Carlo method. Moreover, to our knowledge, there is not any other algorithm capable of achieving this.

More complex systems, like non-holonomic vehicles, can also be studied with this algorithm. The main problem for this case is the amount of time required to perform the guaranteed integration calculations. In our tests, a regular non-holonomic robot with a simple line follower controller takes around 6 hours to complete the calculations using the same accuracy as the one used in the pendulum example.

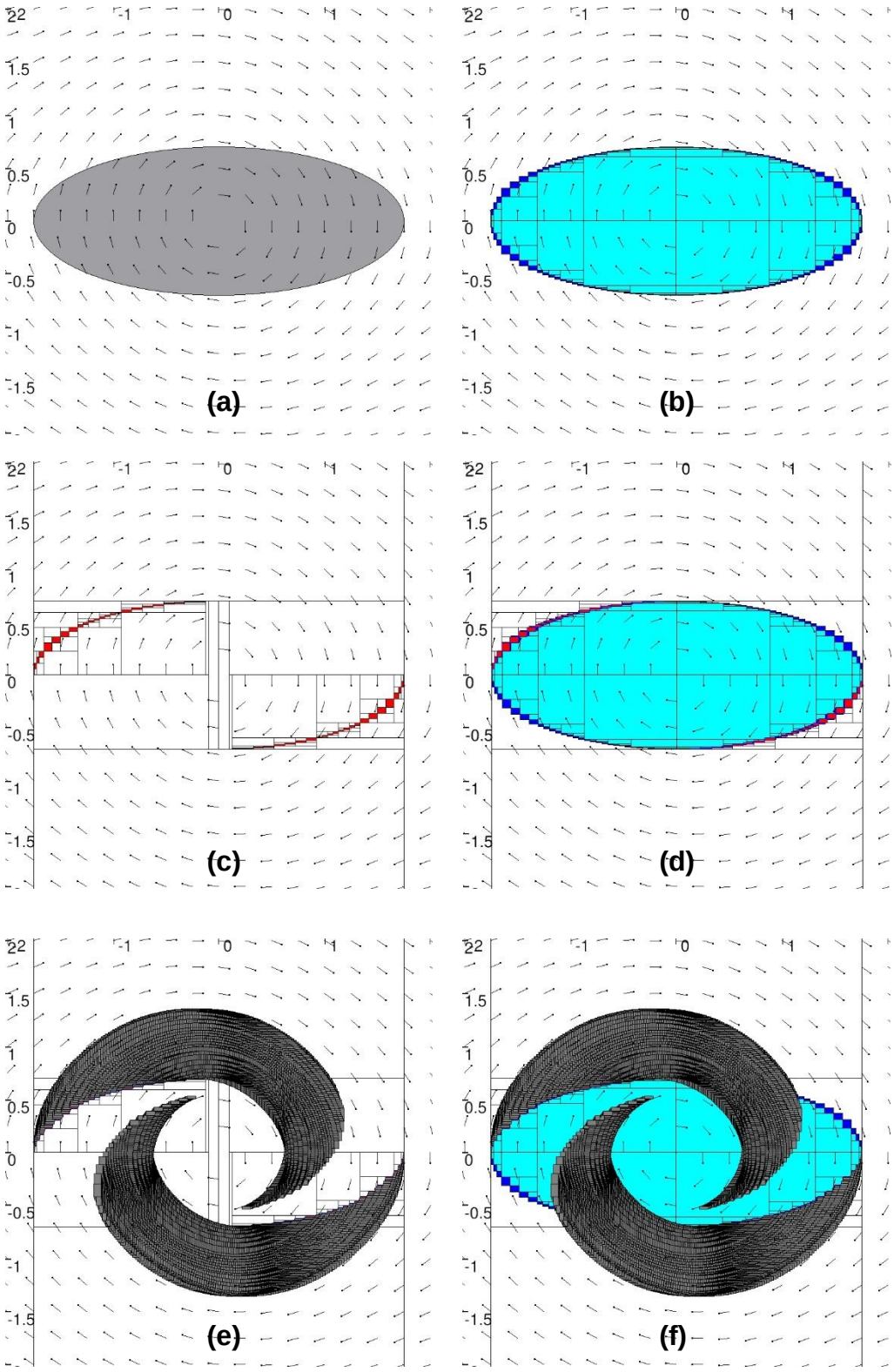

Fig. 13 Algorithm results. Subfigure (a) shows in gray the initial capture tube. Subfigure (b) presents the inner (light blue) and boundary (dark blue) subpaving of the initial capture tube. In subfigure (c) we can find the boxes that contains trajectories that leave the initial tube (red). Subfigure (d) presents all the previous results together. Subfigure (e) shows the guaranteed integration (dark grey) of all the trajectories that leave the tube. Subfigure (f) superposes the previous results to illustrate the outer approximation of $\mathbb{G}$. For all the subfigures, the horizontal axis corresponds to x_1 and the vertical axis to x_2.

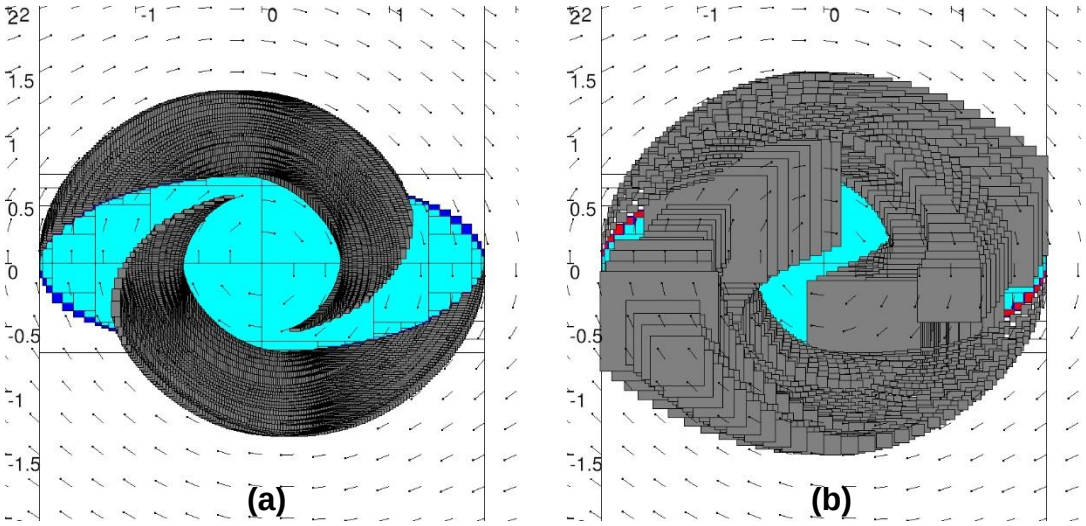

Fig. 14 Comparison of the use wrapping effect obtained from guaranteed integration (Subfigure (a)) and Euler integration (Subfigure (b)) for performing a simulation of a system for a time of three seconds. In the case of the Euler integration, the time step was set to 0.05 *seconds.*

Chapter 6

Conclusions and Future Work

The problem of proving that a controlled nonlinear system stays always inside a time-varying bubble (tube) is equivalent to proving a set of nonlinear inequalities is inconsistent. In practice, even with good intuition and following the existent methods, it is difficult to find such a significant capture tube. In this dissertation we propose that, given a candidate tube $\mathbb{G}(t)$, the developed algorithm can compute an approximation for the smallest capture tube $\mathrm{capt}(\mathbb{G}(t))$, which encloses the candidate $\mathbb{G}(t)$. Because of the system's properties, even when $\mathbb{G}(t)$ is chosen as attractive, it is possible that during the initialization some trajectories can leave $\mathbb{G}(t)$. Moreover, $\mathrm{capt}(\mathbb{G}(t))$ may not be able to be representable in a computer; to solve this, the algorithm characterizes this capture tube by an inner and outer approximation that define an interval of tubes which encloses it. The core of the algorithm is the guaranteed integration of the trajectories that leave the candidate tube $\mathbb{G}(t)$ in order to obtain $\Delta\mathbb{G}(t)$.

G-Stability is a very young field and therefore there are many improvements that can be applied to this algorithm. One of the most important would be to develop a technique to obtain a suitable first approximation of $\mathbb{G}(t)$ for any system. An additional important point to study is the reduction of the use of guaranteed integration in order to improve the calculation efficiency and reduce the pessimism derived from the wrapping effect. This can be done by

using more complex shapes, for instance barrier functions [64], to define $\mathbf{g}(\mathbf{x}, t)$.

The algorithm developed for this dissertation is able to find the smallest capture tube that encloses another tube. Analogously, it is also interesting to find the largest capture tube that belongs to another tube.

Due to the nature of the integration problem, it could also be parallelised using threads [65] or a more specific solution based on graphics processors [66], to improve the computation time.

Additional measures can be taken to further reduce the computational time. These include the calculation of inner solutions for the unsafe boxes using vertex calculations [67]. This will give us a dynamic approximation of time (contrary to the fixed integration time used now) required by each of the system's unsafe boxes to return to the initial tube.

We have already extended our algorithm to find the largest capture tube enclosed by another tube, implemented vertex integration and a method to obtain the initial tube from empirical data. The results of these studies will be presented in the Second International Symposium on Set Membership - Applications, Reliability and Theory (SMART 2015) [14].

References

[1] E. J. Routh, *A Treatise on the Stability of a Given State of Motion: Particularly Steady Motion*. Macmillan, 1877.

[2] H. Nyquist, "Regeneration Theory," *Bell Syst. Tech. J.*, vol. 11, no. 1, pp. 126–147, Jan. 1932.

[3] R. E. Kalman and J. F. Bertram, "Control System Analysis and Design via the Second Method of Lyapunov," *J. Basic Engrg*, vol. 88, 1960.

[4] A. M. Lyapunov, *General Problem of the Stability Of Motion (1892)*, vol. 3. CRC Press, 1992.

[5] T. Khailat, *Linear Systems*. Prentice Hall, 1980.

[6] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[7] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Birkhauser, 2008.

[8] J.-P. Aubin, *Viability Theory*. Birkhauser, 1991.

[9] L. Jaulin and F. Le Bars, "An interval approach for stability analysis: Application to sailboat robotics," *Robot. IEEE Trans.*, vol. 29, no. 1, pp. 282–287, 2012.

[10] A. Stancu, L. Jaulin, and A. Bethencourt, "Stability analysis for time-dependent nonlinear systems. An interval approach," *Submitt. to Autom.*, vol. 44, no. 0, p. 161, 2015.

[11] G. Chabert and L. Jaulin, "Contractor programming," *Artif. Intell.*, vol. 173, pp. 1079–1100, 2009.

[12] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, 1993.

[13] L. Jaulin, D. Lopez, V. Le Doze, S. Le, J. Ninin, G. Chabert, M. Saad, and A. Stancu, "Computing capture tubes," *Reliab. Comput.*, 2015.

[14] D. Lopez, "Computing Capture Tubes," in *SMART 2015 | The University of Manchester | University of Manchester Aerospace Research Institute*, 2015.

[15] R. E. Kalman, "Lyapunov functions for the problem of Lur'e in automatic control," *Proc Nat Acad.Sci USA*, vol. 49, no. 2, pp. 201 – 205, 1963.

[16] J. P. LaSalle and S. Lefschetz, *Stability by Lyapunov's Second Method with Applications*. New York: Academic, 1961.

[17] H. Poincaré, "Sur les courbes definies par les equations differentielles," *Journal de mathématiques pures et appliquées*, 1881.

[18] N. Elia and S. K. Mitter, "Stabilization of linear systems with limited information," *IEEE Trans. Automat. Contr.*, vol. 46, no. 9, pp. 1384–1400, 2001.

[19] M. Rachid, M. Xavier, K. Firas, and O. Alain, "Stability and resonance conditions of elementary fractional transfer functions," *Automatica*, vol. 47, no. 11, pp. 2462–2467, 2011.

[20] M. S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 4, pp. 475–482, Apr. 1998.

[21] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Syst. Mag.*, vol. 19, no. 5, pp. 59–70, 1999.

[22] W. Tucker, "The lorenz attractor exists," *Comptes Rendus l'Acadmie des Sci. - Ser. I - Math.*, vol. 328, no. 12, pp. 1197–1202, 1999.

[23] Y. Lin, E. D. Sontag, and Y. Wang, "A Smooth Converse Lyapunov Theorem for Robust Stability," *SIAM J. Control Optim.*, vol. 34, no. 1, pp. 124–160, Jan. 1996.

[24] J.-B. Pomet and L. Praly, "Adaptive nonlinear regulation: estimation from the Lyapunov equation," *IEEE Trans. Automat. Contr.*, vol. 37, no. 6, pp. 729–740, Jun. 1992.

[25] B. Bolzano, *Paradoxien des Unendlichen*. C.H. Reclam, 1851.

[26] M. Nagumo, "Uber die lage der integralkurven gew¨ ohnlicher differentialgleichungen," *Proc. Phys-Math. Soc. Japan*, vol. 24, no. 3, pp. 272–559, 1942.

[27] J. P. Aubin, "Viability Theory," 1991.

[28] F. H. Clarke, *Optimization and non smooth analysis*. New York: Wiley, 1983.

[29] H. Brezis, "On a characterization of flow-invariant sets," *Comm. Pure Appl. Math*, vol. 23, no. 261–263, pp. 261–263, 1970.

[30] F. L. Bars and L. Jaulin, "An experimental validation of a robust controller with the VAIMOS autonomous sailboat," in *Proceedings of the 5th International Robotic Sailing Conference*, 2012.

[31] R. E. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1966.

[32] R. E. Moore, "Methods and Applications of Interval Analysis," *SIAM*, 1979.

[33] R. L. Muhanna and R. L. Mullen, "Uncertainty in Mechanics Problems—Interval–Based Approach," *J. Eng. Mech.*, vol. 127, no. 6, pp. 557–566, Jun. 2001.

[34] N. Metropolis and S. Ulam, "The Monte Carlo Method," *J. Am. Stat. Assoc.*, vol. 44, no. 247, p. 335, Sep. 1949.

[35] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the Monte Carlo method is so important today," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 6, no. 6, pp. 386–392, Nov. 2014.

[36] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, p. 97, Apr. 1970.

[37] P. Herrero, Z. Chen, J. Bondia, P. Georgiou, C. Toumazou, and A. Schaschkow, "Interval-based model predictive control for an artificial pancreas," *Diabetes Technol. Ther.*, vol. 17, pp. A99–A99, 2015.

[38] J. Aubin and H. Frankowska, *Set-Valued Analysis*. Birkhuser Boston, 1990.

[39] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*. Springer-Verlag, 2001.

[40] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, no. 1, p. 35, 1960.

[41] G. Chabert, "ibex-lib.org," 2007. [Online]. Available: http://www.ibex-lib.org/. [Accessed: 28-Aug-2015].

[42] Q. Brefort, L. Jaulin, M. Ceberio, and V. Kreinovich, "Towards Fast and Reliable Localization of an Underwater Object: An Interval Approach," *J. Uncertain Syst.*, vol. 9, 2015.

[43] M. S. I. Seddik, L. Jaulin, and J. Grimdale, "Phase Based Localization for Underwater Vehicles Using Interval Analysis," *Math. Comput. Sci. Spec. issue Interval methods Appl.*, vol. 8, no. 3, pp. 495–502, 2014.

[44] U. Rauf, T. Kirnicky, and M. Whelan, *Dynamics of Civil Structures, Volume 2: Proceedings of the 33rd IMAC, A Conference and Exposition on Structural Dynamics, 2015*. Springer, 2015.

[45] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*. Providence: American Mathematical Society, 2012.

[46] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, "Validated Solutions of Initial Value Problems for Ordinary Differential Equations," *Appl. Math. Comput.*, vol. 105, no. 1, pp. 21–68, 1999.

[47] J. dit Sandretto and A. Chapoutot, "Validated Solution of Initial Value Problem for Ordinary Differential Equations based on Explicit and Implicit Runge-Kutta Schemes," Jan. 2015.

[48] A. Chapoutot, "DynIBEX," 2014. [Online]. Available: http://perso.ensta-paristech.fr/~chapoutot/dynibex/. [Accessed: 28-Aug-2015].

[49] C. Barbarosie, "Reducing the wrapping effect," *Computing*, vol. 54, no. 4, pp. 347–357, 1995.

[50] R. B. Kearfott, Ed., *Reliable Computing*. Kluwer Academic.

[51] R. E. Moore, R. B. Kearfott, and M. J. Cloud, "Introduction to Interval Analysis," *Soc. Ind. Appl. Math.*, 2009.

[52] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, "Chapter 3," in *Applied Interval Analysis*, Springer-Verlag, 2001.

[53] R. Lohner, "Enclosing the solutions of ordinary initial and boundary value problems," in *Computer Arithmetic: Scientific Computation and Programming Languages*, U. K. E. Kaucher and C. Ullrich, Eds. BG Teubner,Stuttgart, Germany, 1987, pp. 255–286.

[54] T. Kapela and P. Zgliczynski, "A lohner-type algorithm for control systems and ordinary differential inclusions," *Discret. Contin. Dyn. Syst.*, vol. 11, no. 2, pp. 365–385, 2009.

[55] D. Wilczak and P. Zgliczynski, "Cr-lohner algorithm," *Schedae Informaticae*, vol. 20, pp. 9–46, 2011.

[56] The Qt Project, "Index of /archive/qt/4.8/4.8.6," 2014. [Online]. Available: http://download.qt.io/archive/qt/4.8/4.8.6/. [Accessed: 28-Aug-2015].

[57] The Qt Project, "Qt 4.8 Documentation," 2014. [Online]. Available: http://doc.qt.io/qt-4.8/install-win.html. [Accessed: 28-Aug-2015].

[58] The Qt Project, "QFile Class." [Online]. Available: http://doc.qt.io/qt-5/qfile.html. [Accessed: 28-Aug-2015].

[59] The Qt Project, "QList Class." [Online]. Available: http://doc.qt.io/qt-4.8/qlist.html. [Accessed: 28-Aug-2015].

[60] Canonical inc., "Ubuntu 14.04.3 LTS (Trusty Tahr)," 2015. [Online]. Available: http://releases.ubuntu.com/14.04/. [Accessed: 28-Aug-2015].

[61] A. Gleixner, M. Miltenberger, and B. Müller, "SoPlex 1.7.2 download," 2013. [Online]. Available: http://soplex.zib.de/download.php?fname=soplex-1.7.2.tgz. [Accessed: 28-Aug-2015].

[62] V. Drevelle and J. Nicola, "VIBes," 2014. [Online]. Available: http://enstabretagnerobotics.github.io/VIBES/. [Accessed: 28-Aug-2015].

[63] G. Chabert, "Minibex syntax — IBEX 2.1.16 documentation," 2015. [Online]. Available: http://www.ibex-lib.org/doc/tutorial.html#using-the-minibex-syntax. [Accessed: 28-Aug-2015].

[64] O. Bouissou, A. Chapoutot, A. Djaballah, and M. Kieffer, "Computation of parametric barrier functions for Dynamical systems using interval analysis," in *IEEE CDC*, 2014.

[65] K. Wheeler, R. Murphy, and D. Thain, "Qthreads: An API for Programming with Millions of Lightweight Threads," 2008.

[66] J. Fang, A. L. Varbanescu, and H. Sips, *A Comprehensive Performance Comparison of CUDA and OpenCL*. Delft, the Netherlands: Delft University of Technology, 2011.

[67] L. Kolev, "Approximate solution of a transient tolerance problem for linear circuits," *IEEE Trans. CAS*, pp. 1–13, 1993.