UCL Université catholique de Louvain
Faculté des Sciences Appliquées
Département d'Ingénierie Informatique
B-1348 Louvain-la-Neuve, Belgique

A Constraint Satisfaction Approach for Enclosing Solutions to Initial Value Problems for Parametric Ordinary Differential Equations

Micha Janssen

Thèse présentée en vue de l'obtention du grade de
Docteur en Sciences Appliquées

Composition du jury :

Prof. Y. Deville (INGI/UCL) - *Co-promoteur*
Prof. P. van Hentenryck (Brown University, USA) - *Co-promoteur*
Prof. P. Delsarte (INGI/UCL) - *Examinateur*
Dr. R. Lohner (Universität Karlsruhe, Allemagne) - *Examinateur*
Prof. M. Rueher (Université de Nice, France) - *Examinateur*
Prof. P. Van Dooren (INMA/UCL) - *Examinateur*
Prof. M. Lobelle (INGI/UCL) - *Président*

Octobre 2001

Abstract

This work considers initial value problems (IVPs) for ordinary differential equations (ODEs) where some of the data is uncertain and given by intervals as is the case in many areas of science and engineering. Interval methods provide a way to approach these problems but they raise fundamental challenges in obtaining high accuracy and low computation costs. This work introduces a constraint satisfaction approach to these problems which enhances traditional interval methods with a pruning step based on a global relaxation of the ODE. The relaxation uses Hermite interpolation polynomials and enclosures of their error terms to approximate the ODE. Our work also shows how to find an evaluation time for the relaxation that minimizes its local error. Theoretical and experimental results show that the approach produces significant improvements in accuracy over the best interval methods for the same computation costs. The results also indicate that the new algorithm should be significantly faster when the ODE contains many operations.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Yves Deville and Pascal van Hentenryck. Yves gave me the taste for research. Since my last year as an undergraduate student, he has always supported and encouraged me. He very often helped me clarifying my vague initial ideas, provided me with useful advice and constructive comments. Our collaboration was very effective and rewarding.

Pascal was an incredible source of motivation to me. Our exciting discussions always led to new ideas and research directions. Ever since we met, he has shown a constant interest in my work and has offered his entire availability to help me develop my ideas and publish them. In this respect, our collaboration has been extremely successful. Pascal helped me formalizing the important concepts of the work and contributed significantly to my thesis. I would also like to thank him for having invited me twice at Brown University in order to work together on several papers. This was a very pleasurable experience.

I'm grateful to the members of my jury - Philippe Delsarte, Rudolf Lohner, Michel Rueher and Paul Van Dooren - for the interest they showed in my work, their careful reading of my first draft, their interesting comments, and their helpful suggestions to improve the draft. Philippe often gave me very useful advice for some purely mathematical aspects of my work. I also whish to thank Marc Lobelle, as president of my jury.

I further would like to acknowledge the INGI Department and its staff for having hosted me for the duration of my thesis and having allowed me to participate at various conferences all over the world. Many thanks to Freddy Gridelet, Marie-France Declerfayt and Francis Degey for providing a good computing environment and for their technical support. Warm thanks to our secretary, Viviane Dehut, for her unlimited availability and her permanent humour.

I am also thankful to all my colleagues and friends who have made these three years of research a very enjoyable experience. Being afraid of omitting somebody, I will not cite them separately but I am confident that they will easily recognize themselves. Thank you all for the good moments we have spent together here in Louvain-la-Neuve!

My thankfulness further goes to my parents, Anny and Wim, for having allowed me to make exciting studies and having supported me all those years, as well as to my brother, Manu, with whom I have spent so many nice moments.

Finally, I acknowledge the *Université catholique de Louvain* and the *Commu-*

nauté française de Belgique for their financial support, and the Belgian *Fonds National de la Recherche Scientifique* (F.N.R.S.), which offered me a research fellowship.

Micha Janssen

Louvain-la-Neuve, octobre 2001

Contents

List of Tables

List of Figures

List of Symbols

1. Small letters denote $\begin{cases} \text{real values} \\ \text{vectors of real values} \\ \text{functions of real values.} \end{cases}$

2. Capital letters denote $\begin{cases} \text{matrices} \\ \text{sets} \\ \text{intervals} \\ \text{vectors of intervals} \\ \text{functions of intervals.} \end{cases}$

3. $\mathbb{O}$: ordinary differential equation (ODE).

4. $\mathbb{N}$: naturals.

5. $\mathbb{R}$: reals.

6. *Bool* : booleans.

7. $\mathbb{IR}$: set of all intervals.

8. box : interval vector.

9. $\mathbb{IR}^n$: set of all n-dimensional boxes.

10. $\square A$: smallest box containing $A \subseteq \mathbb{R}^n$.

11. $g(A) = \{g(x) \mid x \in A\}$.

12. M^{-1} : enclosure of the inverse of M.

13. $t_i, t_e, t \in \mathbb{R}$.

14. $u_i \in \mathbb{R}^n$.

15. $D_i, B_i \in \mathbb{IR}^n$.

16. $m(D)$: midpoint of the box D.

17. $s(D) = D - m(D)$.

18. $\omega(D)$: width of the box D.

19. $\mathcal{J}_{\tilde{x}} g(x) = \begin{bmatrix} \frac{\partial g_1}{\partial x_{i_1}}(x) & \dots & \frac{\partial g_1}{\partial x_{i_p}}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_{i_1}}(x) & \dots & \frac{\partial g_n}{\partial x_{i_p}}(x) \end{bmatrix}$ where $\tilde{x} = (x_{i_1}, \dots, x_{i_p})$.

20. $\mathcal{J} g(x) = \mathcal{J}_x g(x)$.

21. n : dimension of the ODE (number of scalar equations).

22. h : step size of the integration.

23. k : multistep = k-step.

24. $\begin{cases} \mathbf{a} & = & (a_0, \dots, a_k) \in A^{k+1}, \\ \mathbf{a}_i & = & (a_{ik}, \dots, a_{(i+1)k-1}) \in A^k, \\ \mathbf{a}_{i..i+j} & = & (a_i, \dots, a_{i+j}) \in A^{j+1}. \end{cases}$

25. $f(x) = \begin{cases} O(g(x)) & \textit{if} & \exists c > 0, \exists \varepsilon > 0 : x \geq \varepsilon \Rightarrow |f(x)| \leq c|g(x)|, \\ \mathcal{O}(g(x)) & \textit{if} & \exists c > 0, \exists \varepsilon > 0 : x \leq \varepsilon \Rightarrow |f(x)| \leq c|g(x)|, \\ \Omega(g(x)) & \textit{if} & \exists c > 0, \exists \varepsilon > 0 : x \leq \varepsilon \Rightarrow |f(x)| \geq c|g(x)|, \\ \Theta(g(x)) & \textit{if} & f(x) = \mathcal{O}(g(x)) \ \textit{and} \ f(x) = \Omega(g(x)). \end{cases}$

26. $C^r(\Omega)$: set of all functions whose r-th derivative exists and is continuous on Ω.

27. $s(t_0, u_0, t)$: solution of an ODE.

28. $ms(\mathbf{t}, \mathbf{u}, t)$: multistep solution of an ODE.

29. $p(\mathbf{t}, \mathbf{u}, t)$: approximation of the multistep solution.

30. $e(\mathbf{t}, \mathbf{u}, t) = ms(\mathbf{t}, \mathbf{u}, t) - p(\mathbf{t}, \mathbf{u}, t)$: error in the approximation of the multistep solution.

31. $(x)_j = \frac{1}{j!} \frac{\partial^j s}{\partial t^j}(t_0, x, t)\Big|_{(t_0, x, t_0)}$.

32. $\{(x)_j \mid x \in D\} \subseteq (D)_j \in \mathbb{IR}^n$.

33. $\mathcal{J}(x)_j = \mathcal{J}_x \frac{1}{j!} \frac{\partial^j s}{\partial t^j}(t_0, x, t)\Big|_{(t_0, x, t_0)}$.

34. $\{\mathcal{J}(x)_j \mid x \in D\} \subseteq \mathcal{J}(D)_j \in \mathbb{IR}^{n \times n}$.

35. $(x)_{j,l}$, $(D)_{j,l}$, $\mathcal{J}(x)_{j,l}$, $\mathcal{J}(D)_{j,l}$: respectively the l-th component of $(x)_j$, $(D)_j$, $\mathcal{J}(x)_j$, $\mathcal{J}(D)_j$.

36. $int(A)$: interior of A.

37. D_1^- : predicted box.

38. D_1^* : pruned box.

39. $FL(\mathbf{t}, \mathbf{D}, t)$: multistep filter for ODE.

40. $\sigma = (\sigma_0, \ldots, \sigma_k)$: parameters of a Hermite filter.

41. $\sigma_s = \sum_{i=0}^{k} \sigma_i$.

42. $\sigma_m = \max(\sigma)$.

43. $w(t) = \prod_{i=0}^{k} (t - t_i)^{\sigma_i}$.

44. $\gamma(t) = \sum_{i=0}^{k} \frac{\sigma_i}{t-t_i}$.

45. GHF(σ) : global Hermite(σ) filter.

46. IHO : Nedialkov's [Ned99] <u>I</u>nterval <u>H</u>ermite-<u>O</u>breschkoff method.

47. IHO* : our variant of Nedialkov's IHO method.

48. AWA : Lohner's [Loh87] <u>A</u>nfangs<u>W</u>ert<u>A</u>ufgabe program.

49. N_1 : number of $*, /$ operations in f.

50. N_2 : number of $\pm$ operations in f.

Chapter 1

Introduction

1.1 The Initial Value Problem for Parametric ODEs

Initial value problems (IVPs) for ordinary differential equations (ODEs) arise naturally in many applications in science and engineering, including chemistry, physics, molecular biology, and mechanics to name only a few. An *ordinary differential equation* $\mathbb{O}$ is a system of the form

$$\begin{aligned} {u_1}'(t) &= f_1(u_1(t), \ldots, u_n(t)) \\ &\vdots \\ {u_n}'(t) &= f_n(u_1(t), \ldots, u_n(t)), \end{aligned} \tag{1.1}$$

often denoted in vector notation by $u'(t) = f(u(t))$ or $u' = f(u)$, in the unknown function u of the single independent variable t. [1] Note that such a system is explicit in the derivative u' of the function u. A *parametric* ODE is an ODE where the function f contains parameters. In this work, we talk about ODEs to denote both traditional and parametric ODEs. An *initial value problem* is an ODE with an initial condition $u(t_0) = u_0$. In addition, in practice, it is often the case that the parameters in f and/or the initial value u_0 are not known with certainty but are given as intervals. In general, *analytical* methods fail to solve initial value problems. Therefore, *numerical* methods play a fundamental role in obtaining approximate solutions to these problems. Numerical methods do not try to approximate the solution to an IVP over a continuous range of the variable t but instead over a *discrete* range of time points $t_0, t_1, \ldots, t_m$.

[1] Only autonomous systems are considered in this work. It is easy to generalize the results to non-autonomous systems.

1.2 Standard Numerical Methods

Standard numerical methods produce floating-point approximations of the solutions at different points in time. They include *one-step* and *multistep*, *explicit* and *implicit* methods. Well known methods are the Euler, Taylor, Runge-Kutta, Adams-Moulton-Bashforth and BDF methods [Hen62, Lam91, Sha93]. These methods are usually fast and reliable for many applications, but there are cases where they can produce inaccurate results. Furthermore, when the parameters and/or the initial values of an IVP are not known with certainty but are given as intervals (as is often the case in practical applications), standard numerical methods may not be the simplest way to approach the resulting parametric ordinary differential equations since, in essence, they would have to solve infinitely many systems.

1.3 Validated Methods

Validated or *interval* methods, pioneered by Moore [Moo66], return enclosures of exact solutions at different points in time, i.e., for a given IVP, they are guaranteed to return intervals containing the exact solution. Validated methods for IVPs for ODEs are based on interval arithmetic. They are generally more costly than standard numerical methods. However, they have a number of advantages over the latter:

- They inherently accommodate uncertainty in the parameters or initial values by using intervals instead of floating-point numbers. They are thus well suited for parametric ODEs;
- Since they produce guaranteed enclosures of the exact solutions, they may be very useful for computations that are critical to the safety or reliability of a system;
- They prove numerically the existence and uniqueness of the solution to an IVP for an ODE.

Traditional validated methods usually consist of two processes applied at each integration step: (1) a *bounding box* process that proves existence and uniqueness of the solution and computes a rough enclosure (called a *bounding box*) of the solution over a time interval $[t_0, t_1]$; (2) a *forward* process that computes an enclosure of the solution at t_1. The bounding box process, which is specific to validated methods, is necessary to bound the error terms in the forward process. The forward process is generally realized by applying a one-step Taylor interval method and making extensive use of automatic differentiation [Ral81] to obtain the Taylor coefficients [Eij81, Kru69, Moo66, Moo79]. Lohner's Anfangswertaufgabe (AWA) system [Loh87] was an important step in interval methods which features efficient coordinate transformations to tackle the wrapping effect. More recently, Nedialkov and Jackson's interval Hermite-Obreschkoff (IHO) method [NJ99] improved on AWA by extending a Hermite-

Obreschkoff's approach (which can be viewed as a generalized Taylor method) to intervals. Another recent approach, the Taylor models, was proposed by Berz & Makino [BM98] for reducing the wrapping effect. Their scheme validates existence and uniqueness and also computes tight enclosures of the solution in one process, contrary to the other methods mentioned above.

1.4 Difficulties in Validated Methods

Validated methods encounter a number of important difficulties:

- The need to bound the local error at each step of the integration. The bound may be excessively large or very small step sizes may be required in order to prove numerically that the bound is correct (see Section 3.1);
- The so-called *wrapping effect*, which is the name given to the over-estimation arising when we enclose by a vector of intervals (called a *box*) the exact solution set at a given time. This is a crucial problem which is typical of interval methods. If not handled correctly, the wrapping effect may result in exceedingly large enclosures of the solution after a number of integration steps and it can be shown that the size of the enclosures may grow exponentially, even if the step size converges to zero (see Section 2.4);
- The over-estimations which are inherent to interval computations (see Subsection 2.1.3).

1.5 Introducing Constraints in ODEs

The research described in this work takes a constraint satisfaction approach to ODEs. Its basic idea [DJVH98, JDVH99, JVHD01] is to view the solving of ODEs as the iteration of three processes: (1) a *bounding box* process, (2) a *predictor* process that computes initial enclosures at given times from enclosures at previous times and bounding boxes, and (3) a *pruning* process that reduces the initial enclosures without removing solutions [2]. The real novelty in our approach is the pruning component. It is based on the construction of a non-trivial constraint from a *relaxation* of the ODE, a key concept in constraint satisfaction [VH98]. This constraint can then be used to prune the solution space at the various integration points.

1.6 Contributions

The main contribution of this work is to show that an effective pruning technique can be derived from a relaxation of the ODE, importing a fundamental principle

[2] Observe that interval extensions of predictor/corrector methods (e.g., [NJ99]) can also be viewed as the composition of a predictor and a pruning step.

from constraint satisfaction into the field of validated differential equations. Four main steps are necessary to derive an effective pruning algorithm.

1. The first step consists in obtaining a relaxation of the ODE by safely approximating its solution using Hermite interpolation polynomials;
2. The second step consists in using the mean-value form of this relaxation for more accuracy and efficiency. Unfortunately, these two steps, which were skeched in [JDVH99], are not sufficient and the resulting pruning algorithm still suffers from traditional problems of interval methods;
3. The third fundamental step [JVHD01] consists in globalizing the pruning by considering several successive relaxations together. This idea of generating a global constraint from a set of more primitive constraints is also at the heart of constraint satisfaction. It makes it possible, in this new context, to address the problem of dependencies (and hence the accumulation of errors) and the wrapping effect simultaneously [3];
4. The fourth and final step consists of finding an evaluation time for the relaxation which minimizes the local error of the relaxation. Indeed, the global constraint generated in the third step, being a relaxation of the ODE, is parametrized by an evaluation time. Interestingly, for global filters based on Hermite interpolation polynomials, the (asymptotically) optimal evaluation time is independent from the ODE and induces negligible overhead on the computational cost of the methods.

Theoretical and experimental results show the benefits of the approach. From a theoretical standpoint, the constraint satisfaction approach provides a quadratic improvement in accuracy (asymptotically) over the best interval method we know of for the same computation costs. The theoretical results also show that our approach should be significantly faster for a given precision when the ODE contains many operations. Experimental results, obtained from an object-oriented implementation of our algorithms, confirm the theory. They show that the constraint satisfaction approach often produces significant improvements in accuracy over existing methods for the same computation costs and should produce significant gain in computation times when the ODE contains many operations. Of particular interest is the versatility of the approach which can be tailored to the problem at hand.

1.7 Outline

The thesis is organized as follows. Chapter 2 introduces the main definitions, notations and the necessary background. Chapter 3 presents the state-of-the-art in validated methods for IVPs for ODEs. It also explains the bounding box process, and presents its theoretical basis and methods to implement it. Chapter 4

[3]Global constraints in ordinary differential equations have also been found useful in [CB99]. The problem and the techniques in [CB99] are however fundamentally different.

gives a high-level overview of the constraint satisfaction approach to parametric ODEs. The next four chapters are the core of the work. Chapter 5 introduces *multistep filters*, addresses their problems (i.e. the wrapping effect and an inherent dependency problem) by the notion of *global filters*, and proposes a pruning algorithm based on these global filters. Chapter 6 presents multistep Hermite filters as a special case of multistep filters. Chapter 7 describes how to choose an evaluation time to minimize the local error of a multistep Hermite filter. Chapter 8 presents the overall algorithm. Chapters 9 and 10 report the theoretical and experimental analyses and Chapter 11 concludes the work and proposes some directions for further research. Appendix A reports the benchmarks used in our experimental results. Appendix B analyzes the computational cost of generating Taylor coefficients as well as their Jacobians. Appendix C gives some detailed proofs of results presented in Chapter 7. Finally, Appendix D discusses in some detail the evaluation of Hermite interpolation polynomials.

Chapter 2

Background and Notations

2.1 Interval Arithmetic

2.1.1 Exact Interval Arithmetic

Interval arithmetic was introduced by Moore [Moo66]. An interval is a closed bounded set of real numbers

$$[a, b] = \{x \mid a \leq x \leq b\}. \tag{2.1}$$

We denote by $\mathbb{IR}$ the set of all intervals. Let $I_1 = [a, b] \in \mathbb{IR}, I_2 = [c, d] \in \mathbb{IR}$ and $\circ \in \{+, -, *, /\}$. The interval arithmetic operations are defined by

$$I_1 \circ I_2 = \{x \circ y \mid x \in I_1,\ y \in I_2\} \tag{2.2}$$

where it is assumed that $0 \notin I_2$ if $\circ = /$. These operations can be defined in terms of real arithmetic operations as follows:

$$\begin{aligned}
I_1 + I_2 &= [a + c, b + d], && (2.3)\\
I_1 - I_2 &= [a - d, b - c], && (2.4)\\
I_1 * I_2 &= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}], && (2.5)\\
I_1 / I_2 &= [a, b] * [1/d, 1/c]. && (2.6)
\end{aligned}$$

Algebraic properties of interval arithmetic can be found e.g. in [NJC99]. In the following, we will omit the symbol $*$ and we will not distinguish between the degenerate interval $[a, a]$ and the real number a. Finally, the relation "=" between two intervals I_1 and I_2 is defined by

$$I_1 = I_2 \Leftrightarrow I_1 \cap I_2 \neq \emptyset. \tag{2.7}$$

2.1.2 Rounded Interval Arithmetic

When implementing interval arithmetic on a finite-precision machine, the bounds of an interval are no longer reals, but floating-point numbers. Moreover, the result of an arithmetic operation on two floating-point numbers is not necessarily a floating-point number. However, we are interested in using interval arithmetic for computing rigorous bounds on the solutions of ODEs and it is possible to program interval arithmetic on a computer with appropriate rounding (called *directed* rounding) of left and right computed endpoints of the intervals, so that the computed interval result always contains the exact interval result. This is called *rounded* or *machine* interval arithmetic.

2.1.3 Over-Estimations in Interval Arithmetic

When interval arithmetic is used to bound the range of a function, the computed range is often an important over-estimation of the exact range. This over-estimation of the range of a function is typical of interval computations. let us illustrate this phenomenon by considering the following function:

$$g(x) = x - x. \tag{2.8}$$

The exact range of g is of course the interval $[0, 0]$. Now assume that we want to compute the range of g for $x \in [0, 1]$ by using interval arithmetic. The computed range is given by

$$[0, 1] - [0, 1] = [-1, 1] \tag{2.9}$$

and is thus an important over-estimation of the exact range. The reason for this is that, although the two occurences of x in the expression of g denote the *same* variable, when we replace the variable x by its range $[0, 1]$, the two occurences of the interval $[0, 1]$ are independent in interval arithmetic. In other terms, any point of the first interval is combined with any point of the second interval. This important problem of interval computations arises whenever several occurences of the same variable appear in an expression. Therefore, two equivalent real expressions, e.g. $x(x - 1)$ and $x^2 - x$, will yield different results in interval arithmetic, depending on their syntactic form. Mean-value forms, presented in Section 2.5, often allow to reduce significantly the over-estimations due to interval arithmetic by putting expressions in a particular syntactic form.

2.2 Basic Notational Conventions

Small letters denote real values, vectors and functions of real values. Capital letters denote matrices, sets, intervals, vectors and functions of intervals. A vector of intervals $D \in \mathbb{IR}^n$ is called a *box*. If $A \subseteq \mathbb{R}^n$, then $\Box A$ denotes the smallest box $D \in \mathbb{IR}^n$ such that $A \subseteq D$ and $g(A)$ denotes the set $\{g(x) \mid x \in A\}$. If M is a regular (point or interval) matrix, then M^{-1} denotes an *enclosure* [1] of the inverse of M. A relation is a function $r : \mathbb{R}^n \to Bool$, where $Bool$ denotes the booleans. We also assume that t_i, t_e and t are reals, u_i is in $\mathbb{R}^n$, and D_i and B_i are in $\mathbb{IR}^n$ ($i \in \mathbb{N}$). We use $m(D)$ to denote the midpoint of D and $s(D)$ to denote $D - m(D)$. Observe that $m(D) + s(D) = D$. We use $\omega(D)$ to denote the width of a box. More precisely, $\omega([a, b]) = b - a$ and $\omega((I_1, \ldots, I_n)) = (\omega(I_1), \ldots, \omega(I_n))$

[1] By *enclosure* of a set A, we mean a set containing A.

if $I_i \in \mathbb{IR}$. If $g : \mathbb{R}^m \to \mathbb{R}^n$, $x = (x_1, \ldots, x_m)$ and $\tilde{x} = (x_{i_1}, \ldots, x_{i_p})$ with $i_1, \ldots, i_p \in 1..m$, then $\mathcal{J}_{\tilde{x}} g(x)$ denotes the Jacobian matrix

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_{i_1}}(x) & \cdots & \frac{\partial g_1}{\partial x_{i_p}}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_{i_1}}(x) & \cdots & \frac{\partial g_n}{\partial x_{i_p}}(x) \end{bmatrix}. \tag{2.10}$$

In particular, we write $\mathcal{J}g(x) = \mathcal{J}_x g(x)$ (differentiation wrt all variables of g). If not specified, n denotes the dimension of the ODE (i.e., the number of scalar equations), $h > 0$ denotes the step size of the integration, and k denotes the number of previous values of the solution at times $t_0, \ldots, t_{k-1}$ used to compute the new value at time t_k (k-step approach).

Notation 1 (Bold Face Notations) *Let A be a set and $a_i \in A$ where $i \in \mathbb{N}$. We use the following bold face notations.*

$$\begin{array}{lcl} \mathbf{a} & = & (a_0, \ldots, a_k) \in A^{k+1}, \\ \mathbf{a}_i & = & (a_{ik}, \ldots, a_{(i+1)k-1}) \in A^k, \\ \mathbf{a}_{i..i+j} & = & (a_i, \ldots, a_{i+j}) \in A^{j+1}. \end{array} \tag{2.11}$$

Observe that $\mathbf{a}_0 = (a_0, \ldots, a_{k-1})$, $\mathbf{a}_1 = (a_k, \ldots, a_{2k-1})$, and $\mathbf{a} = (a_0, \ldots, a_k)$. The following asymptotical notations are standard:

Notation 2 (Asymptotical Notations) *Consider two functions $f, g : \mathbb{R} \to \mathbb{R}$ and let $x > 0$. We use the following standard notations.*

$$f(x) = \begin{cases} O(g(x)) & if \;\; \exists c > 0, \exists \varepsilon > 0 : x \geq \varepsilon \Rightarrow |f(x)| \leq c|g(x)|, \\ \mathcal{O}(g(x)) & if \;\; \exists c > 0, \exists \varepsilon > 0 : x \leq \varepsilon \Rightarrow |f(x)| \leq c|g(x)|, \\ \Omega(g(x)) & if \;\; \exists c > 0, \exists \varepsilon > 0 : x \leq \varepsilon \Rightarrow |f(x)| \geq c|g(x)|, \\ \Theta(g(x)) & if \;\; f(x) = \mathcal{O}(g(x)) \;\; and \;\; f(x) = \Omega(g(x)). \end{cases} \tag{2.12}$$

The notations extend component-wise for vectors and matrices of functions.

Finally we assume that the underlying interval arithmetic is exact for the theoretical parts of this work (i.e. there are no rounding errors).

2.3 Basic Definitions

As traditional, when we consider an ODE $u' = f(u)$ and an interval of integration T, we assume $f \in C^r(\Omega)$, where r is sufficiently large and Ω is an open set such that $T \times \Omega$ contains the trajectories of the solutions on T [2]. In addition, we restrict our attention to ODEs that have a unique solution for a given initial value. Techniques to verify this hypothesis numerically are well-known (see Section 3.1). In order to make the dependence on the initial condition (t_0, u_0) explicit, we introduce the following definition of the solution to an ODE.

[2]The standard mathematical symbol $C^r(\Omega)$ denotes the set of all functions whose r-th derivative exists and is continuous on Ω.

Definition 1 (Solution of an ODE) *Let $\Lambda \subseteq \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}$ be an open set. The* solution *of an ODE $u' = f(u)$ is the function $s : \Lambda \to \mathbb{R}^n$ such that*

$$\forall (t_0, u_0, t) \in \Lambda : \begin{cases} \frac{\partial s}{\partial t}(t_0, u_0, t) = f(s(t_0, u_0, t)), \\ s(t_0, u_0, t_0) = u_0. \end{cases} \tag{2.13}$$

Observe that, since we restrict attention to autonomous systems in this work, we can write

$$s(t_0, x, t) = s(0, x, \tau) \tag{2.14}$$

where $\tau = t - t_0$, and thus

$$\frac{\partial^j s}{\partial t^j}(t_0, x, t) = \frac{\partial^j s}{\partial \tau^j}(0, x, \tau). \tag{2.15}$$

In particular, when $t = t_0$, the function

$$\left. \frac{\partial^j s}{\partial t^j}(t_0, x, t) \right|_{(t_0, x, t_0)} = \left. \frac{\partial^j s}{\partial \tau^j}(t_0, x, \tau) \right|_{(0, x, 0)} \tag{2.16}$$

depends *only* on x. This justifies the following notation, which captures the notions of real and interval Taylor coefficients of the solution of an ODE as well as their Jacobians.

Notation 3 (Taylor Coefficients and Jacobians) *Let s be the solution of an ODE $\mathbb{O}$, $x \in \mathbb{R}^n$, $D \in \mathbb{IR}^n$, and let t_0 be any real number. Then,*

1. $(x)_j = \frac{1}{j!} \frac{\partial^j s}{\partial t^j}(t_0, x, t) \Big|_{(t_0, x, t_0)}$;
2. $\{(x)_j \mid x \in D\} \subseteq (D)_j \in \mathbb{IR}^n$;
3. $\mathcal{J}(x)_j = \mathcal{J}_x \frac{1}{j!} \frac{\partial^j s}{\partial t^j}(t_0, x, t) \Big|_{(t_0, x, t_0)}$;
4. $\{\mathcal{J}(x)_j \mid x \in D\} \subseteq \mathcal{J}(D)_j \in \mathbb{IR}^{n \times n}$;
5. *$(x)_{j,l}$, $(D)_{j,l}$, $\mathcal{J}(x)_{j,l}$ and $\mathcal{J}(D)_{j,l}$ denote respectively the l-th component of $(x)_j$, $(D)_j$, $\mathcal{J}(x)_j$ and $\mathcal{J}(D)_j$.*

In the context of our multistep approach (to be presented in Chapter 4), it is useful to generalize Definition 1 in order to make the dependence on the last $k+1$ redundant conditions $(t_0, u_0), \ldots, (t_k, u_k)$ explicit.

Definition 2 (Multistep solution of an ODE) *Let s be the solution of an ODE $\mathbb{O}$. The* multistep solution *of $\mathbb{O}$ is the* partial *function $ms : A \subseteq \mathbb{R}^{k+1} \times (\mathbb{R}^n)^{k+1} \times \mathbb{R} \to \mathbb{R}^n$:*

$$ms(\mathbf{t}, \mathbf{u}, t) = \begin{cases} s(t_0, u_0, t) \text{ if } \ u_i = s(t_0, u_0, t_i), \ 1 \leq i \leq k, \\ \text{undefined otherwise.} \end{cases} \tag{2.17}$$

Since we are dealing with interval methods, we need to introduce the notions of interval extensions of a function and a relation. These notions were introduced in [VHMD97]. However, because the techniques proposed in this work use multistep solutions, which are *partial* functions, it is necessary to generalize the notion of interval extension to partial functions and relations.

Definition 3 (Interval Extension of a Partial Function) *The interval function $G : \mathbb{IR}^n \to \mathbb{IR}^m$ is an* interval extension *of the partial function $g : E \subseteq \mathbb{R}^n \to \mathbb{R}^m$ if*

$$\forall D \in \mathbb{IR}^n : g(E \cap D) \subseteq G(D). \tag{2.18}$$

Definition 4 (Interval Extension of a Partial Relation) *The interval relation $R : \mathbb{IR}^n \to Bool$ is an* interval extension *of the partial relation $r : E \subseteq \mathbb{R}^n \to Bool$ if*

$$\forall D \in \mathbb{IR}^n : (\exists x \in E \cap D : r(x)) \Rightarrow R(D). \tag{2.19}$$

Given interval extensions for the primitive operations $(\exp, \ln, \sqrt{}, \sin, \cos, \ldots)$ and relations, it is possible to obtain interval extensions for composite functions and relations. The *natural interval extension* is the simplest way to extend a real function (or relation) to an interval function (or relation). The natural interval extension is obtained by replacing each real number by its approximation, each real variable by an interval variable, each real arithmetic operation by the corresponding interval arithmetic operation, and each primitive operation or relation by its interval extension. A more advanced interval extension is the so called *mean-value interval extension*, which is presented in Section 2.5.

Finally, we generalize the concept of bounding boxes, a fundamental concept in interval methods for ODEs, to multistep methods. Intuitively, a bounding box encloses all solutions of an ODE going through certain boxes at given times over a given time interval. Bounding boxes are needed to enclose error terms in validated methods for ODEs (see Chapter 6).

Definition 5 (Bounding Box) *Let $\mathbb{O}$ be an ODE system, ms be the multistep solution of $\mathbb{O}$, and $\{t_0, \ldots, t_k\} \subseteq T \in \mathbb{IR}$. A box B is a* bounding box *of $\mathbb{O}$ over T wrt $(\mathbf{t},\mathbf{D})$ if, for all $t \in T$, $ms(\mathbf{t}, \mathbf{D}, t) \subseteq B$.*

2.4 The Wrapping Effect

The *wrapping effect* is a fundamental concept in interval computations and in particular in validated methods for ODEs [Moo66, Loh01, NJ00]. It is the name given to the overestimation that arises from approximating a set by a box. In the context of ODEs, the set of solutions at each integration step is over-approximated by a box. These over-approximations accumulate step after step and may result in an explosion in the sizes of the computed boxes. The wrapping effect in validated methods for ODEs can be illustrated by Moore's [Moo66] classical example:

$$\begin{array}{rcl} u_1' & = & u_2 \\ u_2' & = & -u_1. \end{array} \tag{2.20}$$

The solution of this ODE for an initial condition $u(0) = u_0$ is given by

$$u(t) = A(t)u_0 \tag{2.21}$$

where

$$A(t) = \begin{bmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{bmatrix}. \tag{2.22}$$

Let us consider a box initial condition $u(t_0) \in D_0$. Figure 2.1 shows the rectangle D_0 in the (u_1, u_2)-phase plane. At time t, the *exact* solution set is a rotated rectangle given by

$$S(t) = \{A(t)u_0 \mid u_0 \in D_0\}. \tag{2.23}$$

Since $A(t)$ is a rotation matrix, it follows that the size of the rotated rectangle $S(t)$ is constant for all t.

Now let us integrate this ODE by computing an enclosing box at each integration step. At a time t_1, the smallest box that encloses the set $S(t_1)$ is given by $D_1 = A(t_1)D_0$ (see Figure 2.1). The set $S(t_1)$ is thus over-approximated by the box D_1. Jackson [Jac75] calls D_1 the *wrapping* of the set $S(t_1)$. Then, at a time t_2, we enclose the exact solution set $\{A(t_2 - t_1)u_1 \mid u_1 \in D_1\}$ corresponding to the initial conditions $u(t_1) \in D_1$ by its wrapping, i.e. the box $D_2 = A(t_2 - t_1)D_1$. Thus, the size of the computed boxes increases step after step. Moore shows that this size grows exponentially, even if the step size converges to zero.

The wrapping effect is thus a critical problem, which has to be dealt with in any validated method. The standard solution used in interval methods for ODEs to obtain tighter solution bounds is to choose, at each step, an appropriate local coordinate system to represent the solutions compactly (see Section 3.3).

2.5 The Mean-Value Form

The mean-value form (MVF) of a function plays a fundamental role in interval computations and is derived from the Mean-Value theorem. Mean-value forms have two important advantages:

1. They often produce tighter intervals in the evaluation of the range of a function;

2. They allow for problem linearizations and an easier treatment of the wrapping effect.

Consider a differentiable function $g : \mathbb{R} \to \mathbb{R}$. By the Mean-Value theorem, we can write

$$g(x) = g(m) + g'(\xi)(x - m) \tag{2.24}$$

for some ξ strictly between x and m. The expression (2.24) is called the mean-value form of g. How can (2.24) be useful in interval computations? Consider

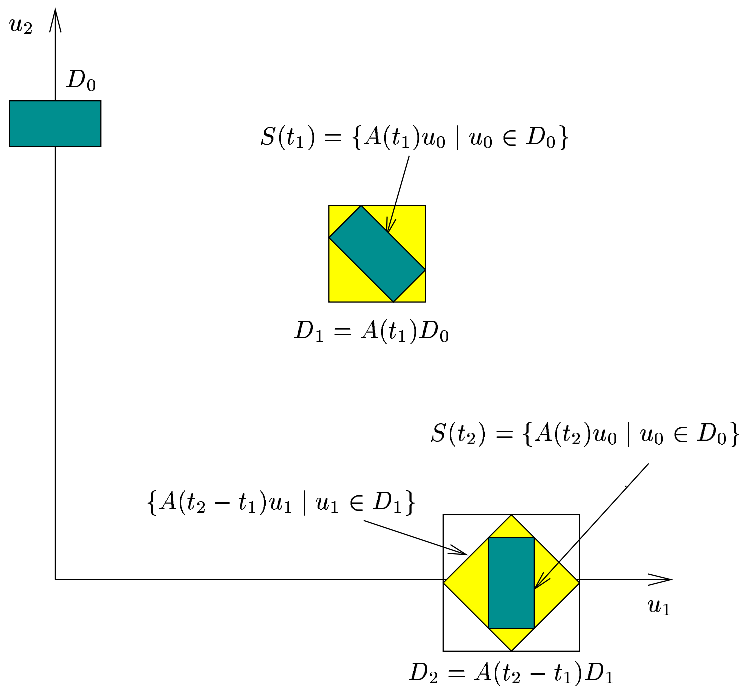
u_2
D_0
$S(t_1) = \{A(t_1)u_0 \mid u_0 \in D_0\}$
$D_1 = A(t_1)D_0$
$S(t_2) = \{A(t_2)u_0 \mid u_0 \in D_0\}$
$\{A(t_2 - t_1)u_1 \mid u_1 \in D_1\}$
u_1
$D_2 = A(t_2 - t_1)D_1$

Figure 2.1: Wrapping Effect in Moore's Example.

an interval I with $x, m \in I$, and let DG be an interval extension of g'. We have the relation

$$g(x) \in g(m) + DG(I)(x - m). \tag{2.25}$$

As a consequence, the interval function

$$G_{I,m}(X) = g(m) + DG(I)(X - m) \tag{2.26}$$

is an interval extension of function g, called the *mean-value interval extension* of g. It can be shown [NJC99] that the mean-value interval extension is effective when the size of the interval I is sufficiently small. Indeed, the over-estimation in the computed bounds is $\mathcal{O}(\omega(I)^2)$. We can of course generalize the above concepts to n-ary functions.

Example

Let $g(x) = x^2 - x$. We want to bound the range of g over the interval $I = [0, 1]$. The first method consists of evaluating the expression of g directly in interval arithmetic by replacing x in g by the interval $[0, 1]$. We obtain:

$$[0, 1]^2 - [0, 1] = [-1, 1]. \tag{2.27}$$

Now let us consider the mean-value form of g. We have $g'(x) = 2x - 1$ and an interval extension of g' can be obtained as $DG(X) = 2X - 1$. Let us choose

$m = m(I)$. The interval extension (2.25) of g is given here by

$$-0.25 + (2[0,1] - 1)(X - 0.5). \tag{2.28}$$

To obtain a bound on the range of g over $I = [0,1]$, we replace X by $[0,1]$ in (2.28):

$$-0.25 + (2[0,1] - 1)([0,1] - 0.5) = [-0.75, 0.25]. \tag{2.29}$$

We have thus obtained a better bound on the range of g than in the direct evaluation of the expression of g in interval arithmetic.

2.6 The Midpoint Technique

The midpoint technique is a standard tool in interval computation. It consists of decomposing a matrix A as the sum of its midpoint matrix and the remainder matrix composed of symmetric intervals:

$$A = m(A) + s(A). \tag{2.30}$$

In this work, the midpoint technique will be very useful in the following two cases:

1. Enclosing a set of real matrix-matrix-vector products (see sections 5.4 and 5.5);
2. Converting an implicit interval linear system into an explicit one by matrix inversion (see subsections 3.4.2 and 5.2.3).

1. Assume that we are interested in enclosing the set

$$P = \{\mathcal{A}\mathcal{B}d \mid \mathcal{A} \in A,\ \mathcal{B} \in B,\ d \in D\} \tag{2.31}$$

where A, B are interval matrices and D is an interval vector. Assume also that $\omega(A)$ is small and that the wrapping effect in the product CD, where $C = AB$, is small. A straightforward and cheap way to enclose the set P consists of computing the product $A(BD)$. But in general, this product does not yield accurate results because of the wrapping effect (see Section 2.4) which occurs two times : (1) in the product $E = BD$ and (2) in the product AE. Another sraightforward way of enclosing the set P is to compute the product $(AB)D$. By hypothesis, the wrapping effect is small in this case and the product is an accurate enclosure of P. However, the multiplication of the two interval matrices A and B is a costly process (due to costly sign tests and rounding mode switches in modern RISC architectures - see [Knu94] for more details). In order to avoid this product, we apply the midpoint technique on A. By distribution and rearrangement of the parentheses, we can write

$$P \subseteq Q = (m(A)B)D + s(A)(BD). \tag{2.32}$$

It is interesting to observe that no multiplication between two interval matrices occurs in Q (note the importance of the parentheses!). From an accuracy standpoint, the wrapping effect in $(m(A)B)D$ is small (by hypothesis) and the remainder term $s(A)(BD)$ is small (because $\omega(A)$ is small). Hence, Q is an accurate enclosure of the set P which avoids the costly multiplication of two interval matrices.

2. Consider the implicit interval linear system

$$\begin{aligned} A_0X_0 + A_1X_1 &= B, \\ X_0 \subseteq D_0, \ X_1 &\subseteq D_1, \end{aligned} \tag{2.33}$$

where A_0, A_1 are interval matrices and B, D_0, D_1 are interval vectors. We assume that A_0 contains no singular point matrix. The exact solution set to this system is given by

$$S = \{(x_0, x_1) \in (D_0, D_1) \mid \exists \mathcal{A}_0 \in A_0, \ \exists \mathcal{A}_1 \in A_1, \ \exists b \in B : \mathcal{A}_0 x_0 + \mathcal{A}_1 x_1 = b\}. \tag{2.34}$$

We are interested in converting the system (2.33) into a system

$$X_0 = CX_1 + E \tag{2.35}$$

which is explicit in the variable X_0 and such that

$$S \subseteq \{(x_0, x_1) \in (D_0, D_1) \mid \exists \mathcal{C} \in C, \ \exists e \in E : x_0 = \mathcal{C}x_1 + e\}. \tag{2.36}$$

A straightforward solution consists of computing an enclosure A_0^{-1} of the inverse of A_0, multypling both sides of (2.33) by A_0^{-1} and rearranging the parentheses:

$$X_0 = -(A_0^{-1}A_1)X_1 + A_0^{-1}B. \tag{2.37}$$

However, the system (2.37) suffers from two drawbacks:

- We have to invert the interval matrix A_0. Computing an accurate enclosure of the inverse of an interval matrix is a costly process (NP hard in general, see [Ned99]);
- We have to multiply the two interval matrices A_0^{-1} and A_1 (see Point 1).

In order to avoid this interval matrix inversion and this product of two interval matrices, we apply the midpoint technique both on A_0 and A_1 in (2.33). By distribution, we can write

$$m(A_0)X_0 = -m(A_1)X_1 + B - s(A_0)X_0 - s(A_1)X_1. \tag{2.38}$$

Since $X_0 \subseteq D_0$ and $X_1 \subseteq D_1$, we can replace X_0 by D_0 in the term involving $s(A_0)$ and X_1 by D_1 in the term involving $s(A_1)$:

$$m(A_0)X_0 = -m(A_1)X_1 + B - s(A_0)D_0 - s(A_1)D_1. \tag{2.39}$$

To obtain a system which is explicit in the variable X_0, we compute an *enclosure* $m(A_0)^{-1}$ of the inverse of the *point* matrix $m(A_0)$, we multiply both sides of (2.39) by $m(A_0)^{-1}$ and we rearrange the parentheses [3]:

$$X_0 = -(m(A_0)^{-1}m(A_1))X_1 + m(A_0)^{-1}(B - s(A_0)D_0 - s(A_1)D_1). \tag{2.40}$$

Observe that in this last system, there is neither an interval matrix inversion nor a product of two interval matrices.

2.7 Automatic Differentiation

In this section, we briefly introduce the technique of automatic differentiation, which in the context of validated methods for ODEs is used to generate Taylor coefficients and Jacobians. More details about automatic differentiation can be found e.g. in [Moo66, Moo79, Ral80, Ral81, Cor88, Cor95]. ***Automatic differentiation is based on well-known differentiation rules for arithmetic and primitive operations, and on the chain rule for composite functions.*** In the following, the operations $+, -, *, /, \exp, \ln, \sqrt{\ }, \sin, \cos, \ldots$ denote either floating-point operations (if we want to compute approximations), or interval operations (if we want to compute enclosures). If v is a function in t, we use the notation

$$(v)_j = \frac{v^{(j)}(t_0)}{j!} \tag{2.41}$$

for the Taylor coefficients of v at the point t_0.

2.7.1 Taylor Coefficients of a Function

Here we show how to compute Taylor coefficients of a function. Let v, w and x be functions in t. For the arithmetic operations, we have the following elementary rules:

$$\begin{aligned}
(v \pm w)_j &= (v)_j \pm (w)_j, && (2.42)\\
(vw)_j &= \sum_{l=0}^{j} (v)_l (w)_{j-l}, && (2.43)\\
(v/w)_j &= (1/w)\left((v)_j - \sum_{l=1}^{j} (w)_l (v/w)_{j-l}\right). && (2.44)
\end{aligned}$$

Differentiation rules for the primitive operations $\exp, \ln, \sqrt{\ }, \sin, \cos, \ldots$ can be obtained from the elementary rules (2.42-2.44) by using the chain rule for composite functions which is given as

$$(v \circ w)'(t) = v'(w(t))w'(t) \tag{2.45}$$

[3]Note that, even though $m(A_0)$ is a *point* matrix, the enclosure $m(A_0)^{-1}$ of its inverse is generally *not* a point matrix, because of rounding errors.

and the following important relation

$$(v)_j = \frac{1}{j}(v')_{j-1}. \tag{2.46}$$

which is a direct consequence of the notation (2.41). As a first example, let us derive the rule for $w = \exp(v)$. By the chain rule (2.45), we can write

$$w' = wv'. \tag{2.47}$$

From (2.46) and the rule (2.43), we have successively

$$\begin{aligned}
(w)_j &= \frac{1}{j}(w')_{j-1} && (2.48)\\
&= \frac{1}{j}(wv')_{j-1} && (2.49)\\
&= \frac{1}{j}\sum_{l=0}^{j-1}(w)_l(v')_{j-1-l} && (2.50)\\
&= \frac{1}{j}\sum_{l=0}^{j-1}(j-l)(w)_l(v)_{j-l}. && (2.51)
\end{aligned}$$

Let us also derive the rules for $w = \cos(v)$ and $x = \sin(v)$. By the chain rule (2.45), we can write

$$\begin{aligned}
w' &= -xv' && (2.52)\\
x' &= wv'. && (2.53)
\end{aligned}$$

From (2.46) and the rules (2.42)(2.43), we have successively

$$\begin{aligned}
(w)_j &= \frac{1}{j}(w')_{j-1} && (2.54)\\
&= \frac{1}{j}(-xv')_{j-1} && (2.55)\\
&= -\frac{1}{j}\sum_{l=0}^{j-1}(x)_l(v')_{j-1-l} && (2.56)\\
&= -\frac{1}{j}\sum_{l=0}^{j-1}(j-l)(x)_l(v)_{j-l}, && (2.57)\\
(x)_j &= \frac{1}{j}(x')_{j-1} && (2.58)\\
&= \frac{1}{j}(wv')_{j-1} && (2.59)\\
&= \frac{1}{j}\sum_{l=0}^{j-1}(w)_l(v')_{j-1-l} && (2.60)
\end{aligned}$$

$$= \frac{1}{j}\sum_{l=0}^{j-1}(j-l)(w)_l(v)_{j-l}. \tag{2.61}$$

Thus, the rules (2.57) and (2.61) for cos and sin have to be used pairwise. From the above rules for arithmetic and primitive operations, we can differentiate to an arbitrary order any function which is a composition of these operations, by using the chain rule (2.45).

2.7.2 Taylor Coefficients of a Solution of an ODE

Now we show how we can compute Taylor coefficients of a solution u of an ODE $u' = f(u)$. By the relation (2.46), we can write

$$\begin{aligned}(u)_j &= \frac{1}{j}(u')_{j-1} &(2.62)\\ &= \frac{1}{j}(f\circ u)_{j-1}. &(2.63)\end{aligned}$$

From (2.63), the Taylor coefficients of u can be generated recursively, i.e., we have to compute the coefficient $(u)_{j-1}$ before we can compute $(u)_j$. From the rules for arithmetic and primitive operations, we can compute the coefficient $(u)_j$ by using the coefficient $(u)_{j-1}$ and the chain rule (2.45).

Example

The following example is taken from [Cor95]. Let us consider the well-known Lorenz system

$$\begin{aligned}x' &= \sigma(y-x) &(2.64)\\ y' &= -xz+\rho x-y &(2.65)\\ z' &= xy-\beta z. &(2.66)\end{aligned}$$

The recursive relation (2.63) is given here by

$$\begin{aligned}(x)_{j+1} &= \sigma((y)_j-(x)_j)/(j+1) &(2.67)\\ (y)_{j+1} &= \left(\rho(x)_j-(y)_j-\sum_{l=0}^{j}(x)_l(z)_{j-l}\right)/(j+1) &(2.68)\\ (z)_{j+1} &= \left(\sum_{l=0}^{j}(x)_l(z)_{j-l}-\beta(z)_j\right)/(j+1). &(2.69)\end{aligned}$$

Chapter 3

Existing Validated Methods

In this chapter, we briefly review the most significant validated methods for IVPs for ODEs. More details can be found e.g. in [Moo66, Loh87, NJC99, Ned99, Rih94]. Most validated methods consist of two processes applied at each step of the integration:

1. A *bounding box* process that computes a step size $h_1 = t_1 - t_0$, proves existence and uniqueness of the solution and computes a bounding box over $[t_0, t_1]$, i.e. a rough a priori enclosure of the range of the solution over $[t_0, t_1]$;

2. A *forward* process that computes a tighter enclosure at t_1 using the latter bounding box.

The intuition of the successive integration steps is illustrated in Figure 3.1.
In Section 3.1, we describe the bounding box process. The following sections are concerned with the forward process. Section 3.3 presents interval Taylor series methods (ITS). Several variants of ITS have been proposed and the most significant one is probably Lohner's method [Loh87]. In Section 3.4, we present

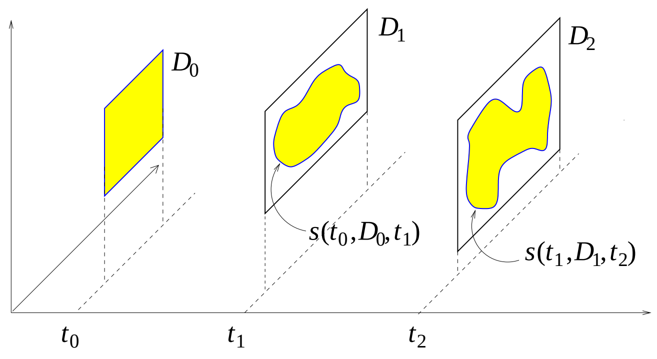

Figure 3.1: Successive Integration Steps.

Nedialkov's recent interval Hermite-Obreschkoff method (IHO). Finally, Section 3.5 describes two other recent approaches, namely Berz & Makino's Taylor Models [BM98] and our piecewise method [DJVH98].

3.1 The Bounding Box Process

3.1.1 The Problem

The bounding box process solves the following problem. Let $f \in C^{p-1}(\Omega)$ where $p \geq 2$. Given a box $D_0 \subseteq \Omega$ and a time t_0, find a step size $h_1 > 0$ and a box B_1 such that for any $u_0 \in D_0$, there exists a unique solution u to the IVP

$$u' = f(u), \quad u(t_0) = u_0 \tag{3.1}$$

over the interval $[t_0, t_1]$, where $t_1 = t_0 + h_1$, and B_1 is a bounding box of $u' = f(u)$ over $[t_0, t_1]$ wrt (t_0, D_0).

To our knowledge, three methods have been proposed to solve this problem: the constant enclosure method, Moore's Taylor series enclosure method [Moo66] (which is a generalization of the first method) and Lohner's polynomial enclosure method [Loh95]. All three methods are based on the Banach fixed-point theorem and the Picard-Lindelöf operator. In this text, we will only present the constant and the Taylor series enclosure methods.

3.1.2 Theoretical Basis

The fundamental theoretical basis of the bounding box methods presented hereafter is the Banach fixed-point theorem which is given as follows.

Theorem 1 (Banach Fixed-Point Theorem) *Let X be a complete non-empty metric space with a metric d and consider an application $\phi : X \rightarrow X$. Let $\alpha \in \mathbb{R}$ satisfy $0 \leq \alpha < 1$. If*

$$\forall x, y \in X : d(\phi(x), \phi(y)) \leq \alpha d(x, y) \tag{3.2}$$

then there exists a unique $x \in X$ such that $x = \phi(x)$.

How can we use this theorem to prove numerically the existence of a unique solution to the IVP (3.1)? If f and u are continuous, then (3.1) is equivalent to the system

$$u = \phi(u) \tag{3.3}$$

where ϕ denotes the Picard-Lindelöf operator defined by

$$(\phi(u))(t) = u_0 + \int_{t_0}^{t} f(u(s))ds. \tag{3.4}$$

The idea is to apply the Banach fixed-point theorem to the Picard-Lindelöf operator and to an appropriate set of functions X. From the assumption $f \in$

$C^{p-1}(\Omega)$ where $p \geq 2$, it can be shown that there is a unique solution $u \in C^p$ to the IVP (3.1) in a neighborhood of t_0 for any $u_0 \in D_0$. Using this result, we can then prove the following theorem, due to Corliss & Rihm [CR96],

Theorem 2 *Let $u_0 \in int(B_1) \subseteq \Omega$ and $f \in C^q(\Omega)$ where $q = \max(1, p-1)$. Consider the interval valued function $U : [t_0, t_1] \to \mathbb{IR}^n$ defined by* [1]

$$U(t) = \sum_{j=0}^{p-1} (t - t_0)^j (u_0)_j + (t - t_0)^p (B_1)_p. \tag{3.5}$$

If

$$\forall t \in [t_0, t_1] : U(t) \subseteq B_1 \tag{3.6}$$

then there is a unique solution u to the IVP (3.1) in $[t_0, t_1]$ and

$$\forall t \in [t_0, t_1] : u(t) \in U(t). \tag{3.7}$$

If the relation (3.6) holds, then B_1 is a bounding box of the ODE over $[t_0, t_1]$ wrt (t_0, u_0). We are now in a position to present a generic bounding box algorithm based on Theorem 2.

3.1.3 A Generic Algorithm

Consider an interval function $\Phi : \mathbb{R} \times \mathbb{IR}^n \to \mathbb{IR}^n$ such that the relation

$$\begin{array}{c} \forall t \in [t_0, t_1], u_0 \in D_0, B_1 \subseteq \Omega : \\ \sum_{j=0}^{p-1} (t - t_0)^j (u_0)_j + (t - t_0)^p (B_1)_p \subseteq \Phi(h_1, B_1) \end{array} \tag{3.8}$$

holds. Assume that $\Phi(h_1, B_1) \subseteq B_1$. From Theorem 2, B_1 is a bounding box of the ODE over $[t_0, t_1]$ wrt (t_0, D_0). Since

$$\begin{array}{rcl} \sum_{j=0}^{p-1} (t - t_0)^j (u_0)_j + (t - t_0)^p (\Phi(h_1, B_1))_p & \subseteq & \Phi(h_1, \Phi(h_1, B_1)) \\ & \subseteq & \Phi(h_1, B_1) \end{array} \tag{3.9}$$

it follows that $\Phi(h_1, B_1)$, $\Phi(h_1, \Phi(h_1, B_1))$, $\Phi(h_1, \Phi(h_1, \Phi(h_1, B_1)))$, ... are also (tighter) bounding boxes of the ODE over $[t_0, t_1]$ wrt (t_0, D_0). From these results, we can build a generic algorithm parametrized by Φ:

1. Guess a step size h_1 and a box B_1;
2. If $\Phi(h_1, B_1) \subseteq B_1$, then we have succeeded and B_1, $\Phi(h_1, B_1)$, $\Phi(h_1, \Phi(h_1, B_1))$, ... can be chosen as a bounding box;
3. Else, try again with another h_1 and B_1.

Several strategies can be applied to guess successive values for h_1 and B_1. One possible technique consists of (1) repeatedly widening the initial guess for B_1 while the inclusion in 2. is not verified; (2) reducing the step size h_1 if the repeated inflations of B_1 still do not yield a validated enclosure.

In the next two subsections, we present the constant and the Taylor series enclosure methods for the bounding box process, which correspond to two instantiations of the interval function Φ in the generic algorithm.

[1] *int*(A) denotes the interior of A.

3.1.4 The Constant Enclosure Method

The constant enclosure method is obtained by defining the function Φ as

$$\Phi(h_1, B_1) = D_0 + [0, h_1]F(B_1) \tag{3.10}$$

where F is an interval extension of f. The function Φ satisfies the relation (3.8) for $p = 1$. Note that Theorem 2 requires that f be in $C^1(B_1)$. As long as f is a composition of $+, -, *, /, \exp, \ln, \sqrt{}, \sin, \cos, \ldots$, i.e. f contains only arithmetic and standard operations, then f is continuous wherever it is defined. As a consequence, the hypothesis $f \in C^1(B_1)$ can be verified numerically simply by evaluating the Jacobian $\mathcal{J}(B_1)_1$. Note also that the forward process usually computes $\mathcal{J}(B_1)_1$ to bound the truncation error. Therefore, this evaluation need not be performed in the bounding box process.

The constant enclosure method is simple but has the disavantage of restricting the step size to Euler steps.

3.1.5 The Taylor Series Enclosure Method

Moore's Taylor series enclosure method [Moo66] is a generalization of the constant enclosure method. It is obtained by defining the function Φ as

$$\Phi(h_1, B_1) = \sum_{j=0}^{p-1} [0, h_1]^j (D_0)_j + [0, h_1]^p (B_1)_p. \tag{3.11}$$

The interval Taylor coefficients $(D_0)_j$ and $(B_1)_p$ in (3.11) can be computed by automatic differentiation (see Section 2.7). Assuming that f contains only elementary and arithmetic operations, the hypothesis $f \in C^{p-1}(B_1)$ where $p \geq 2$ can be verified numerically by evaluating $(B_1)_p$. The Taylor series enclosure method allows larger step sizes than the constant enclosure method. The step size can increase as the order of the series increases. Algorithms implementing the Taylor series enclosure method are proposed e.g. in [Ned99, CR96].

3.2 The Forward Process

3.2.1 The Problem

Consider the IVP (3.1) and let u be its solution. Given

- a box D_{i-1} and a time t_{i-1}, computed at the previous integration step,
- a step size $h_i = t_i - t_{i-1}$ and a bounding box B_i of the ODE over $[t_{i-1}, t_i]$ wrt (t_{i-1}, D_{i-1}), computed in the bounding box process of the current integration step,

the objective is to compute a box D_i containing the exact solution $u(t_i)$ for any $u_{i-1} \in D_{i-1}$, by means of the bounding box B_i.

3.2.2 The Methods

Most methods implementing the forward process are based on Taylor series plus remainder. Section 3.3 presents the most significant ones. Another approach, based on the Hermite-Obreschkoff relation, is presented in Section 3.4. In all these methods, the local truncation error is enclosed using a bounding box.

3.3 Interval Taylor Series Methods

3.3.1 Moore's Taylor Method

The first and simplest method was proposed by Moore [Moo66]. By Taylor's theorem, we can write

$$u_l(t_i) = \sum_{j=0}^{p-1} h_i^j (u_{i-1})_{j,l} + \frac{h_i^p}{p!} u_l^{(p)}(\xi_l) \tag{3.12}$$

where $t_{i-1} < \xi_l < t_i$, $1 \leq l \leq n$. Moore's Taylor method computes a box D_i containing the exact solution $u(t_i)$ by the formula

$$D_i = \sum_{j=0}^{p-1} h_i^j (D_{i-1})_j + h_i^p (B_i)_p. \tag{3.13}$$

This formula however suffers from two important drawbacks:

1. The widths of D_i always increase with i, even if the true solutions actually contract. This comes from the following property of interval addition:
$$\omega([a,b] + [c,d]) = \omega([a+c, b+d]) \geq \omega([a,b]); \tag{3.14}$$
2. The wrapping effect (see Subsection 2.4) is not handled, often leading to quick size explosions of the computed boxes D_i.

To avoid these problems, Moore does not apply his method to compute enclosures for the original IVP, but uses it to compute enclosures for another related IVP, for which the solution changes slowly. Enclosures for the original IVP are then obtained by a local coordinate transformation [Moo66].

3.3.2 Mean-Value Taylor Method

The mean-value Taylor method is based on the mean-value form of a Taylor series (see Section 2.5). By the Mean-Value theorem, (3.12) can be rewritten as

$$u_l(t_i) = \sum_{j=0}^{p-1} h_i^j (m_{i-1})_{j,l} + \left(\sum_{j=0}^{p-1} h_i^j \mathcal{J}(\mu_l)_{j,l} \right) (u_{i-1} - m_{i-1}) + \frac{h_i^p}{p!} u_l^{(p)}(\xi_l) \tag{3.15}$$

where $t_{i-1} < \xi_l < t_i$ and μ_l is on the straight line between m_{i-1} and u_{i-1}, $1 \le l \le n$. We can compute a box $D_i \ni u(t_i)$ by the formula

$$D_i = \sum_{j=0}^{p-1} h_i^j (m(D_{i-1}))_j + \left(\sum_{j=0}^{p-1} h_i^j \mathcal{J}(D_{i-1})_j \right) (D_{i-1} - m(D_{i-1})) + h_i^p (B_i)_p. \tag{3.16}$$

As pointed out in Section 2.5, the mean-value Taylor method is effective when the boxes D_i remain sufficiently small throughout the integration. In that case, it will yield more accurate results than Moore's Taylor method. However, the computational cost of the mean-value Taylor method is $O(n^3)$, whilst the cost of Moore's Taylor method is only $O(n^2)$. It is interesting to observe that the mean-value Taylor method allows for decreasing widths of the successive boxes when the true solutions contract. But it does not handle the wrapping effect occuring in the evaluation of the interval matrix/vector product

$$\left(\sum_{j=0}^{p-1} h_i^j \mathcal{J}(D_{i-1})_j \right) (D_{i-1} - m(D_{i-1})). \tag{3.17}$$

However, it is easy to incorporate local coordinate transformations in (3.16) because the formula is in linear form.

3.3.3 Coordinate Transformations in the Mean-Value Taylor Method

In order to reduce the wrapping effect occuring in the evaluation of the interval expression (3.17), we can make use of local coordinate transformations at each step of the integration. The basic idea is to compute an enclosing box, not in the original coordinate system, but in another system which changes from step to step. This allows for more compact enclosures of the exact solution set at each time t_i. At the i-th step of the integration, the formula (3.16) can be rewritten as

$$D_i = A_i(D_{i-1} - m(D_{i-1})) + K_i \tag{3.18}$$

where

$$A_i = \sum_{j=0}^{p-1} h_i^j \mathcal{J}(D_{i-1})_j, \tag{3.19}$$

$$K_i = \sum_{j=0}^{p-1} h_i^j (m(D_{i-1}))_j + h_i^p (B_i)_p. \tag{3.20}$$

At each step i, we define a local coordinate system by the affine transformation

$$u_i = M_i y_i + c_i \tag{3.21}$$

where $M_i \in \mathbb{R}^{n \times n}$ is a regular matrix and $c_i \in \mathbb{R}^n$. We then compute two boxes D_i and Y_i, respectively in the original and local coordinate systems, as

$$\begin{aligned} D_i &= (A_i M_{i-1})Y_{i-1} + A_i(c_{i-1} - m(D_{i-1})) + K_i, & (3.22) \\ Y_i &= (M_i^{-1}(A_i M_{i-1}))Y_{i-1} + (M_i^{-1}A_i)(c_{i-1} - m(D_{i-1})) \\ &\quad + M_i^{-1}(K_i - c_i). & (3.23) \end{aligned}$$

Note that D_i is used to compute the components of the formula (3.23) at step $i+1$, while Y_i is used to *propagate* the solution via the formula (3.23) (see also Section 5.5). The methods of Eijgenraam [Eij81], Krückeberg [Kru69], Lohner [Loh87] and Rihm [Rih94] correspond to different variants of the formulae (3.22)(3.23). How do we choose the matrix M_i? For simplicity, assume that A_i is a point matrix. We can observe that if the matrix $M_i^{-1}(A_i M_{i-1})$ is diagonal, then no wrapping effect occurs in the evaluation of $(M_i^{-1}(A_i M_{i-1}))Y_{i-1}$. From a theoretical standpoint, an ideal choice is thus $M_i = A_i M_{i-1}$. This choice is the basis of the methods of Krückeberg [Kru69] and Eijgenraam [Eij81]. However, it suffers from serious numerical problems. Indeed, after a number of steps, the matrix $A_i M_{i-1}$ often becomes ill-conditioned. As a consequence, if we try to enclose its inverse, the resulting interval matrix contains very large intervals. Therefore, the method can break down very quickly. In the next section, we describe Lohner's choice for the matrix M_i, which is generally much more efficient numerically. Finally, it is important to note that the associativity (determined by the parentheses) of the multiplications in (3.23) is critical in reducing the wrapping effect.

3.3.4 Lohner's Method

Lohner's method [Loh87] is actually a particular case of (3.22)(3.23), where we take $c_i = m(D_i)$ and M_i is an orthogonal matrix obtained by computing a QR-factorization of a permutation of the matrix $m(A_i)M_{i-1}$. We obtain the formulae

$$\begin{aligned} D_i &= (A_i M_{i-1})Y_{i-1} + K_i, & (3.24) \\ Y_i &= (M_i^{-1}(A_i M_{i-1}))Y_{i-1} + M_i^{-1}(K_i - m(D_i)). & (3.25) \end{aligned}$$

The method is initialized by [2]

$$\begin{aligned} Y_0 &= D_0 - m(D_0), & (3.26) \\ A_0 &= I. & (3.27) \end{aligned}$$

Since Y_0 is a symmetric box, it follows from (3.24)(3.25) that $m(D_i) = m(K_i)$ and Y_i is a symmetric box, for all $i \geq 0$. Let us now explain Lohner's strategy for computing a coordinate transformation matrix M_i.

[2]The standard matrix symbol I denotes the identity matrix.

Orthogonal Transformation Matrices

Let $A_i^* = m(A_i M_{i-1})$. The objective is to enclose the parallelepiped

$$S_i = \{A_i^* y_{i-1} \mid y_{i-1} \in Y_{i-1}\} \tag{3.28}$$

by a compact parallelepiped

$$P_i = \{M_i y_i \mid y_i \in (M_i^{-1} A_i^*) Y_{i-1}\}. \tag{3.29}$$

As mentioned in the previous subsection, the optimal enclosing parallelepiped is $P_i = S_i$, for which we have $M_i = A_i^*$, but in general this leads to numerical instability. Hence, we need to look for a more stable scheme in our choice for the transformation matrix M_i. Lohner's idea is to restrict attention to *orthogonal* matrices. The justification for this restriction is that orthogonal matrices are very well-conditioned. As a result, the enclosing parallelpipeds are not optimal anymore but the much better numerical stability of this scheme compared to the optimal enclosing parallelepiped scheme makes the former generally more efficient than the latter. The remaining issue is the choice of a suitable orthogonal matrix.

An Example

We first illustrate the method on the following example where $n = 2$:

$$\begin{aligned} A_i^* &= \begin{bmatrix} 4 & 2 \\ 6 & 1 \end{bmatrix}, \\ Y_{i-1} &= ([-0.5, 0.5], [-0.5, 0.5]). \end{aligned} \tag{3.30}$$

Note that for $n = 2$, a parallelepiped P_i corresponding to an orthogonal matrix M_i is actually a rotated rectangle in the phase plane. Figure 3.2 (a) shows the parallelepiped S_i and Figure 3.2 (b) shows the box $A_i^* Y_{i-1}$, which is the smallest enclosing box of S_i (i.e. the smallest enclosing parallelepiped P_i of S_i for which $M_i = I$). As can be seen from the figure, this box significantly over-approximates the set S_i.

Lohner suggests that the matrix M_i be chosen such that one edge of the rectangle P_i be parallel to the *longest* edge of the parallelepiped S_i. The resulting rectangle P_i fits well the parallelepiped S_i, as shown in Figure 3.2 (d). Figure 3.2 (c) shows the box $(M_i^{-1} A_i^*) Y_{i-1}$ in the new coordinate system induced by M_i. Figures 3.2 (e) and (f) correpsond to a rotation matrix M_i for which one edge of the rectangle P_i is parallel to the *shortest* edge of the parallelepiped S_i. As can be observed from Figure 3.2 (f), P_i does not fit S_i as well as in Figure 3.2 (d). Consequently, the longest edge heuristic proposed by Lohner appears to be a better one than the shortest edge heuristic.

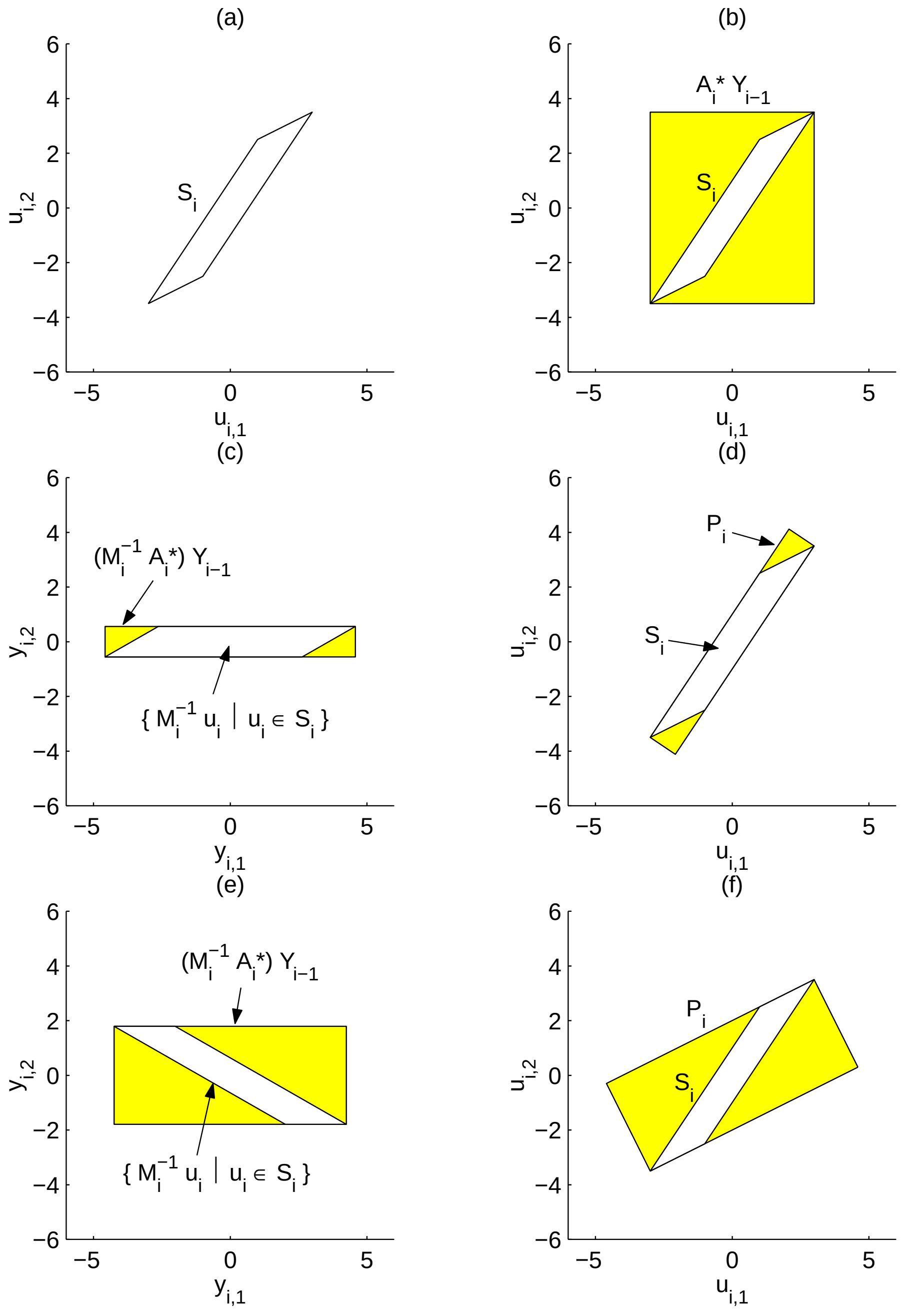
(a)
S_i
$u_{i,1}$
$u_{i,2}$
(b)
$A_i^* Y_{i-1}$
S_i
(c)
$(M_i^{-1} A_i^*) Y_{i-1}$
$\{ M_i^{-1} u_i \mid u_i \in S_i \}$
$y_{i,1}$
$y_{i,2}$
(d)
P_i
S_i
(e)
$(M_i^{-1} A_i^*) Y_{i-1}$
$\{ M_i^{-1} u_i \mid u_i \in S_i \}$
(f)
P_i
S_i

Figure 3.2: Choosing a Local Coordinate System.

The QR-Factorization Method

We now explain how Lohner integrates the above ideas in his QR-factorization technique to compute the matrix M_i. Let $col_1, \dots, col_n$ be the columns of the matrix A_i^*. We can write:

$$A_i^* y_{i-1} = col_1 y_{i-1,1} + \dots + col_n y_{i-1,2}. \tag{3.31}$$

From (3.28), we have

$$S_i = \{col_1 y_{i-1,1} + \dots + col_n y_{i-1,n} \mid y_{i-1} \in Y_{i-1}\}. \tag{3.32}$$

Thus, each edge of S_i is parallel to a column col_j, is generated by the corresponding interval $Y_{i-1,j}$ and has a length

$$l_j = \|col_j\|_2 \omega(Y_{i-1,j}) \quad (1 \leq j \leq n). \tag{3.33}$$

Lohner first rearranges the columns of A_i^* in decreasing order of the values l_j. The resulting matrix can be written as $A_i^* P$ where P is an appropriate permutation matrix. For our example, we have

$$\begin{array}{l} l_1 = \sqrt{4^2 + 6^2}\ 1 = \sqrt{52}, \\ l_2 = \sqrt{2^2 + 1^2}\ 1 = \sqrt{5}. \end{array} \tag{3.34}$$

Since $l_1 > l_2$, the matrix A_i^* remains unchanged and P is the identity matrix. If we had $l_1 < l_2$, then the two columns of A_i^* would be swapped and we would have

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{3.35}$$

The idea is then to perform a QR-factorization [Atk88] of the matrix $A_i^* P$:

$$QR = A_i^* P \tag{3.36}$$

where Q is an orthogonal matrix and R an upper-triangular matrix, and to choose $M_i = Q$ as a coordinate transformation matrix. Let $Z_{i-1} = P^T Y_{i-1}$ and observe that Z_{i-1} is a permutation of the intervals composing Y_{i-1} such that the components of Z_{i-1} generate edges of S_i of decreasing lengths. Since a permutation matrix P is always orthogonal, i.e. $P^{-1} = P^T$, we can write:

$$\begin{array}{rcl} P_i & = & \{M_i y_i \mid y_i \in (M_i^{-1} A_i^*) Y_{i-1}\} \\ & = & \{M_i y_i \mid y_i \in (M_i^{-1} A_i^* P) Z_{i-1}\} \\ & = & \{Q y_i \mid y_i \in (Q^T A_i^* P) Z_{i-1}\} \\ & = & \{Q y_i \mid y_i \in R Z_{i-1}\}, \\ & & \\ S_i & = & \{A_i^* y_{i-1} \mid y_{i-1} \in Y_{i-1}\} \\ & = & \{Q R P^T y_{i-1} \mid y_{i-1} \in Y_{i-1}\} \\ & = & \{Q y_i \mid y_i \in \{R z_{i-1} \mid z_{i-1} \in Z_{i-1}\}\}. \end{array} \tag{3.37}$$

Since R is a triangular matrix, it follows that

- An edge of P_i generated by the interval $Z_{i-1,1}$ is parallel to the longest edge of S_i (which is parallel to the first column of Q);

- An edge of P_i generated by the interval $Z_{i-1,2}$ is parallel to the projection of the second longest edge of S_i on the $(n-1)$-dimensional space that is orthogonal to the first column of Q;

- An edge of P_i generated by the interval $Z_{i-1,3}$ is parallel to the projection of the third longest edge of S_i on the $(n-2)$-dimensional space that is orthogonal to the two first columns of Q;

- And so on.

In practice, we perform the QR-factorization in usual floating-point arithmetic. Because of rounding errors, the matrix M_i is thus a floating-point approximation to an orthogonal matrix. Therefore, starting from the transpose of the matrix M_i, we enclose M_i^{-1} in interval arithmetic. Lohner's QR-factorization technique is currently the best general scheme for computing a coordinate transformation matrix M_i.

3.3.5 Order of an ITS Method

It can be shown (see e.g. [Ned99]) that the overestimation in the local error term $h_i^p(B_i)_p$ in an ITS method is $\mathcal{O}(h_i^{p+1})$, provided that $\omega(B_i) = \mathcal{O}(h_i)$. As a consequence, an ITS method with parameter p, denoted ITS(p), is of order p (since the order of a method is the order of the local error minus 1).

3.4 Interval Hermite-Obreschkoff Method

Recently, a new approach was proposed for computing a tight enclosure in the forward process, namely Nedialkov and Jackson's interval Hermite-Obreschkoff (IHO) method [NJ99, Ned99]. The Taylor series based methods presented in the previous section are explicit methods as the computation of the enclosures simply consists in the evaluation of an interval Taylor series together with a bound on the associated error term. Instead, the IHO method is implicit as it requires to solve a nonlinear system of equations. In standard numerical analysis, Hermite-Obreschkoff methods are usually interesting for stiff problems (note that implicit methods are known to be more appropriate for stiff problems). The IHO method however is not targeted specifically to stiff problems but is designed to be a general-purpose method that has a better precision for a smaller computation cost than the explicit validated methods based on Taylor series.

3.4.1 The Hermite-Obreschkoff Relation

We first present the Hermite-Obreschkoff relation, which is the basis of the IHO method. Let

$$c_j^{q,p} = \frac{q!}{(p+q)!} \frac{(q+p-j)!}{(q-j)!} \tag{3.38}$$

where $p, q, j \geq 0$. It can be shown that the following relation holds:

$$\sum_{j=0}^{q} (-1)^j c_j^{q,p} h_1^j (u_1)_{j,i} = \sum_{j=0}^{p} c_j^{p,q} h_1^j (u_0)_{j,i} + (-1)^q \frac{q!p!}{(p+q)!} \frac{h_1^{p+q+1}}{(p+q+1)!} u_i^{(p+q+1)}(\xi_i) \tag{3.39}$$

where $t_0 < \xi_i < t_1$, $1 \leq i \leq n$. It is interesting to observe that the Hermite-Obreschkoff relation (3.39) is actually a *generalization* of both the explicit and implicit Taylor series relations:

- If $p > 0$ and $q = 0$, then (3.39) becomes an *explicit* Taylor series relation:

$$u_i(t_1) = \sum_{j=0}^{p} h_1^j (u_0)_{j,i} + \frac{h_1^{p+1}}{(p+1)!} u_i^{(p+1)}(\xi_i); \tag{3.40}$$

- If $p = 0$ and $q > 0$, then (3.39) becomes an *implicit* Taylor series relation:

$$u_i(t_0) = \sum_{j=0}^{q} (-1)^j h_1^j (u_1)_{j,i} + (-1)^{q+1} \frac{h_1^{q+1}}{(q+1)!} u_i^{(q+1)}(\xi_i). \tag{3.41}$$

Note that Rihm [Rih98] proposed an interval method based on (3.41).

3.4.2 The IHO Method

The IHO method computes an enclosure of the solution set at t_1 in two phases, denoted as *predictor* and *corrector* by Nedialkov [Ned99]:

1. *Predictor*: compute a first (coarse) enclosure of the solution set at t_1 using an interval Taylor series method of order $q + 1$;

2. *Corrector*: tighten this enclosure by enclosing the solution of the Hermite-Obreschkoff relation (3.39).

The Predictor

The predictor used in IHO is based on a mean-value Taylor method. Given a matrix M_{i-1} and a box Y_{i-1} in the coordinate system induced by M_{i-1}, both computed at the previous step $i - 1$ in the corrector (see below), the predictor computes a box D_i^0 by the formula

$$D_i^0 = (A_i M_{i-1}) Y_{i-1} + K_i, \tag{3.42}$$

where

$$\begin{aligned} A_i &= \sum_{j=0}^{q} h_i^j \mathcal{J}(D_{i-1})_j, && (3.43) \\ K_i &= \sum_{j=0}^{q} h_i^j (m(D_{i-1}))_j + h_i^{q+1}(B_i)_{q+1}. && (3.44) \end{aligned}$$

The Corrector

The first step in the derivation of the IHO method consists of considering the mean-value form of the Hermite-Obreschkoff relation (3.39). By the Mean-Value theorem, (3.39) can be rewritten as

$$\begin{aligned} & c\sum_{j=0}^{q}(-1)^j c_j^{q,p} h_1^j (m_1)_{j,i} + \left(\sum_{j=0}^{q}(-1)^j c_j^{q,p} h_1^j \mathcal{J}(\mu_i)_{j,i}\right)(u_1 - m_1) \\ & \quad = \sum_{j=0}^{p} c_j^{p,q} h_1^j (m_0)_{j,i} + \left(\sum_{j=0}^{p} c_j^{p,q} h_1^j \mathcal{J}(\nu_i)_{j,i}\right)(u_0 - m_0) \\ & \qquad + (-1)^q \frac{q!p!}{(p+q)!}\frac{h_1^{p+q+1}}{(p+q+1)!} u_i^{(p+q+1)}(\xi_i) \qquad (3.45) \end{aligned}$$

where $t_0 < \xi_i < t_1$, μ_i is on the straight line between m_1 and u_1, and ν_i is on the straight line between m_0 and u_0, $1 \leq i \leq n$. Let us define the following quantities:

$$\begin{aligned} A_i^- &= \sum_{j=0}^{q}(-1)^j c_j^{q,p} h_i^j \mathcal{J}(D_i^0)_j, && (3.46) \\ A_i^+ &= \sum_{j=0}^{p} c_j^{p,q} h_i^j \mathcal{J}(D_{i-1})_j, && (3.47) \\ K_i &= \sum_{j=0}^{p} c_j^{p,q} h_i^j (m(D_{i-1}))_j - \sum_{j=0}^{q}(-1)^j c_j^{q,p} h_i^j (m(D_i^0))_j \\ &\quad + (-1)^q \frac{q!p!}{(p+q)!} h_i^{p+q+1}(B_i)_{p+q+1}. && (3.48) \end{aligned}$$

At the i-th integration step, an interval extension of the relation (3.45) is given by

$$A_i^-(D_i - m(D_i^0)) = A_i^+(D_{i-1} - m(D_{i-1})) + K_i. \qquad (3.49)$$

This relation is a linear interval system of equations in the variable D_i. We would like to invert the matrix A_i^- to make (3.49) explicit in the variable D_i. However, as pointed out by Nedialkov [Ned99], computing a tight enclosure of the inverse of an *interval* matrix is NP hard in general. For this reason, we first apply the midpoint technique (see Section 2.6) to (3.49), so that we only need to enclose the inverse of a *point* matrix. We obtain:

$$m(A_i^-)(D_i - m(D_i^0)) = A_i^+(D_{i-1} - m(D_{i-1})) - s(A_i^-)(D_i^0 - m(D_i^0)) + K_i. \quad (3.50)$$

By inverting the matrix $m(A_i^-)$, we obtain the interval relation

$$\begin{aligned} D_i - m(D_i^0) &= (m(A_i^-)^{-1}A_i^+)(D_{i-1} - m(D_{i-1})) \\ &-(m(A_i^-)^{-1}s(A_i^-))(D_i^0 - m(D_i^0)) + m(A_i^-)^{-1}K_i. \end{aligned} \quad (3.51)$$

In practice, $m(A_i^-)^{-1}$ is not a point matrix but an enclosure of the inverse of $m(A_i^-)$, because of rounding errors. Several techniques can be used to compute $m(A_i^-)^{-1}$, e.g. the method of Rump [Rum92, Rum93]. Note that in the IHO method, the interval matrix $m(A_i^-)^{-1}s(A_i^-) = m(A_i^-)^{-1}(A_i^- - m(A_i^-))$ is replaced with $m(A_i^-)^{-1}A_i^- - I$. The equation (3.51) is thus transformed into

$$D_i - m(D_i^0) = G_i(D_{i-1} - m(D_{i-1})) + H_i(D_i^0 - m(D_i^0)) + m(A_i^-)^{-1}K_i \quad (3.52)$$

where

$$\begin{aligned} G_i &= m(A_i^-)^{-1}A_i^+, & (3.53) \\ H_i &= I - m(A_i^-)^{-1}A_i^-. & (3.54) \end{aligned}$$

The last step in the derivation of the IHO method consists of introducing coordinate transformations in the formula (3.52). At each integration step i, we define a local coordinate system by the affine transformation

$$u_i = M_i y_i + m(D_i). \quad (3.55)$$

We then compute two boxes D_i and Y_i, respectively in the original and local coordinate systems, as

$$\begin{aligned} D_i &= \big[m(D_i^0) + (G_i M_{i-1})Y_{i-1} + H_i(D_i^0 - m(D_i^0)) \\ &\quad + m(A_i^-)^{-1}K_i\big] \cap D_i^0, & (3.56) \\ Y_i &= (M_i^{-1}(G_i M_{i-1}))Y_{i-1} + (M_i^{-1}H_i)(D_i^0 - m(D_i^0)) \\ &\quad + (M_i^{-1}m(A_i^-)^{-1})K_i + M_i^{-1}(m(D_i^0) - m(D_i)). & (3.57) \end{aligned}$$

Note that D_i is used to compute the components of the formula (3.57) at step $i+1$, while Y_i is used to *propagate* the solution via the formula (3.57) (see also Section 5.5). The intersection with the predicted box D_i^0 in (3.56) guarantees that the corrector never produces a wider enclosure than D_i^0. The IHO method uses Lohner's QR-factorization technique to compute a coordinate transformation matrix M_i.

3.4.3 Order of the IHO Method

Nedialkov shows that the overestimation in the term $h_i^{p+q+1}(B_i)_{p+q+1}$ is $\mathcal{O}(h_i^{p+q+2})$, provided that $\omega(B_i) = \mathcal{O}(h_i)$ (see [Ned99]). As a consequence, the IHO method with parameters p, q, denoted IHO(p, q), is of order $p + q + 1$ (since the order of a method is the order of the local error minus 1).

3.4.4 Reuse of Taylor coefficients and Jacobians from the Predictor

Finally, it is important to note that the quantities

$$h_i^j \mathcal{J}(D_{i-1})_j, \quad j = 1, \dots, q, \tag{3.58}$$

$$h_i^j (m(D_{i-1}))_j, \quad j = 1, \dots, q, \tag{3.59}$$

occuring respectively in the formulas (3.47) and (3.48) have already been computed in the predictor (respectively in the formulas (3.43) and (3.44)), and they are thus not recomputed in the corrector as long as we have $p \leq q$. Nedialkov claims that usually a good choice for q is $q \in \{p, p+1, p+2\}$ [Ned99].

3.4.5 Comparison with ITS Methods

Nedialkov and Jackson have shown [NJ99, Ned99] that for the same order and step size, the IHO method has smaller local truncation error, better stability, and requires fewer Jacobian evaluations than an ITS method (which is in general the most costly part of the methods). Consider an ITS(r) and the IHO(p, q) methods. Let $p = q$ and $r = p+q+1$. For simplicity, we assume that (the natural encoding of) function f contains only arithmetic operations. We denote by N_1 the number of $*, /$ operations in f, and by N_2 the number of $\pm$ operations. In ITS, we generate $r-1$ Jacobians $\mathcal{J}(D_{i-1})_1, \dots, \mathcal{J}(D_{i-1})_{r-1}$. From Appendix B, the number of interval arithmetic operations needed to generate these $r-1$ Jacobians is given by:

$$(r-1)^2 n N_1 + O(rnN_2). \tag{3.60}$$

In IHO, we generate $p = (r-1)/2$ Jacobians $\mathcal{J}(D_{i-1})_1, \dots, \mathcal{J}(D_{i-1})_p$ and $q = p = (r-1)/2$ Jacobians $\mathcal{J}(D_i)_1, \dots, \mathcal{J}(D_i)_q$. The corresponding number of interval arithmetic operations is thus given by:

$$\frac{(r-1)^2}{2} n N_1 + O(rnN_2). \tag{3.61}$$

Thus, an ITS method requires approximately twice as many operations as the IHO method of the same order to generate the Jacobians. The extra cost in the corrector in the IHO method is one matrix inversion and a few matrix multiplications. In general, compared to an ITS method, the IHO method produces tighter enclosures in less time.

3.5 Other Approaches

3.5.1 Taylor Models

Recently, Berz and Makino [BM98] proposed a method, called the Taylor models, for reducing the wrapping effect. Their method is based on high-order Taylor series expansions with respect to both time and the initial conditions.

The arithmetic operations and standard functions take Taylor polynomials as operands. Berz and Makino's scheme validates existence and uniqueness, and computes a tight enclosure of the solution set at each integration time t_i in one process (contrary to the ITS and IHO methods). Note that this is similar to Lohner's polynomial enclosure approach [Loh95].

3.5.2 The Piecewise Method

In [DJVH98, DJVH98a], we proposed the piecewise method for advancing the solution from t_0 to t_1. Its basic idea is to take an interval extension $S(t_0, D_0, t_1)$ of the solution $s(t_0, u_0, t_1)$ to an ODE, and to both minimize and maximize each component of the function S on the domain D_0. These $2n$ optimization problems can be solved using consistency techniques as in the NUMERICA system [VHMD97]. The interval function S can be realized by means of any existing interval method, e.g. an ITS method. We also showed how to integrate local coordinate transformations in our piecewsie method to reduce the wrapping effect. See [DJVH98, DJVH98a] for more details about the piecewise method.

3.5.3 Comparison with ITS and IHO Methods

The Taylor models approach requires arithmetic with Taylor polynomials, which makes it significantly more expensive computationally than ITS or IHO methods. The piecewise method requires to solve $2n$ optimization problems at each step of the integration and is also significantly more costly than ITS or IHO methods. However, the Taylor models and the piecewise methods can become interesting when the size of the initial conditions (i.e. of the box D_0) is very large. In that case, the IHO method and efficient ITS methods such as Lohner's method will perform badly because they are based on mean-value forms, which require that the initial conditions be of sufficiently small size (see Section 2.5). Since the Taylor models and the piecewise method are not based on mean-value forms, they will generally be successful for large interval initial conditions. In that sense, they are orthogonal to the ITS and IHO methods.

Chapter 4

The Constraint Satisfaction Approach

The constraint satisfaction approach followed in this work was first presented in [DJVH98]. It consists of a generic algorithm for ODEs that iterates three processes:

1. A *bounding box* process that computes bounding boxes for the current step and proves (numerically) the existence and uniqueness of the solution;
2. A *predictor* process that computes initial enclosures at given times from enclosures at previous times and bounding boxes;
3. A *pruning* process that reduces the initial enclosures without removing solutions.

Bounding box and predictor components are standard in interval methods for ODEs. This work thus focuses on the pruning process, the main novelty of the approach.

Our pruning component is based on relaxations of the ODE, a fundamental concept in the field of constraint satisfaction.

To our knowledge, no other approach uses relaxations of the ODE to derive pruning operators and the only other approaches using a pruning component [NJ99, Rih98] were developed independently. Note also that, in the following, predicted boxes are generally superscripted with the symbol $-$ (e.g., D_1^-), while pruned boxes are generally superscripted with the symbol $*$ (e.g., D_1^*).

4.1 Constraints in Nonlinear Programming

The pruning component uses *safe approximations* of the ODE to shrink the boxes computed by the predictor process. To understand this idea, it is useful to contrast the constraint satisfaction to nonlinear programming [VHMK97, VHMD97] and to ordinary differential equations. In nonlinear programming, a constraint $c(x_1, \dots, x_n)$ can be used almost directly for pruning the search space (i.e., the Cartesian product of the intervals I_i associated with the variables x_i). It suffices to take an interval extension $C(X_1, \dots, X_n)$ of the constraint. Now if $C(I'_1, \dots, I'_n)$ does not hold, it follows, by definition of interval extensions, that no solution of c lies in $I'_1 \times \dots \times I'_n$. The interval extension can be seen as a filter that can be used for pruning the search space in many ways. For instance, Numerica uses box(k)-consistency on these interval constraints [VHMD97]. Ordinary differential equations raise new challenges. In an ODE $\forall\ t:\ u' = f(u)$, functions u and u' are, of course, unknown. Hence it is not obvious how to obtain a filter to prune boxes.

4.2 Constraints in ODEs

One of the main contributions of our approach is to show how to derive effective pruning operators for parametric ODEs. The first step consists in rewriting the ODE $\forall\ t:\ u' = f(u)$, in terms of its multistep solution ms to obtain

$$\forall\ t:\ \frac{\partial ms}{\partial t}(\mathbf{t}, \mathbf{u}, t) = f(ms(\mathbf{t}, \mathbf{u}, t)). \tag{4.1}$$

Let us denote this relation $\forall\ t : fl(\mathbf{t}, \mathbf{u}, t)$. This rewriting may not appear useful since ms is still an unknown function. However it suggests a way to approximate the ODE. Indeed, we show in Chapter 6 how to obtain interval extensions of ms and $\frac{\partial ms}{\partial t}$ by using Hermite polynomial interpolations together with their error terms. This simply requires a bounding box for the considered time interval and safe approximations of ms at successive times, both of which are available from the bounding box and predictor processes. Once these interval extensions are available, it is possible to obtain an interval relation of the form

$$\forall\ t : FL(\mathbf{t}, \mathbf{D}, t) \tag{4.2}$$

which approximates the original ODE *safely* in the sense that, if $FL(\mathbf{t}, \mathbf{D}, t)$ does not hold for a time t, it follows that no solution of the ODE can go through boxes $D_0, \dots, D_k$ at times $t_0, \dots, t_k$ (we say that the relation FL is sound). The above relation is still not ready to be used as a filter because t is universally quantified. The solution here is simpler and consists of restricting attention to a finite set T of times (possibly a singleton) to obtain the relation

$$\forall\ t\ \in\ T : FL(\mathbf{t}, \mathbf{D}, t) \tag{4.3}$$

which produces a computable filter. The relation FL is a *relaxation* of the ODE (4.1) in a constraint satisfaction sense [VH98], i.e., given a time t, it produces a relation that can be used to prune the domain of the variables. The so-obtained relation is in fact a conservative approximation of the actual ODE at the given time. The following definition and proposition capture these concepts more formally.

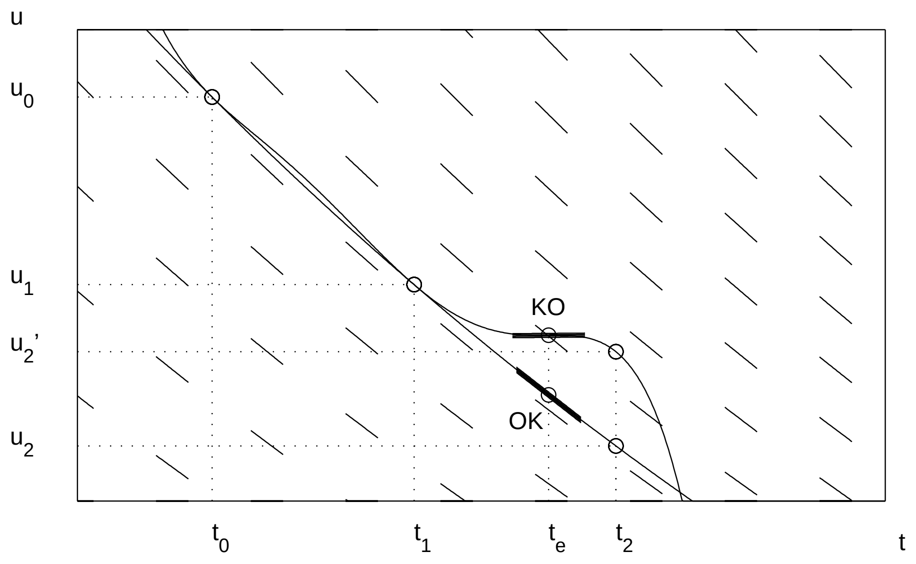

Figure 4.1: Geometric Intuition of the Multistep Filter.

Definition 6 (Multistep Filter) *Let $\mathbb{O}$ be an ODE and s its solution. A* multistep filter *for $\mathbb{O}$ is an interval relation $FL : \mathbb{R}^{k+1} \times (\mathbb{IR}^n)^{k+1} \times \mathbb{R} \to Bool$ satisfying*

$$\left.\begin{array}{c} u_i \in D_i \\ s(t_0, u_0, t_i) = u_i \ (0 \leq i \leq k) \end{array}\right\} \Rightarrow \ \forall t : \ FL(\mathbf{t}, \mathbf{D}, t). \tag{4.4}$$

The variable t is called the evaluation time *of the multistep filter.*

Proposition 1 (Soundness of Multistep Filters) *Let $\mathbb{O}$ be an ODE and let FL be a multistep filter for $\mathbb{O}$. If $FL(\mathbf{t}, \mathbf{D}, t)$ does not hold for some t, then there exists no solution of $\mathbb{O}$ going through $\mathbf{D}$ at times $\mathbf{t}$.*

How can we use this filter to obtain tighter enclosures of the solution? A simple technique consists of pruning the last box computed by the predictor process. Assume that D_i^* is a box enclosing the solution at time t_i $(0 \leq i < k)$ and that we are interested in pruning the last predicted box D_k^-. A subbox $D \subseteq D_k^-$ can be pruned away if the condition

$$FL(\mathbf{t}, (D_0^*, \ldots, D_{k-1}^*, D), t_e) \tag{4.5}$$

does not hold for some evaluation point t_e. Let us explain briefly the geometric intuition behind this relation by considering what we call *natural filters*. Given interval extensions MS, DMS and F respectively of ms, $\frac{\partial ms}{\partial t}$ and f, it is possible to approximate the ODE $u' = f(u)$ by the relation

$$DMS(\mathbf{t}, \mathbf{D}, t) = F(MS(\mathbf{t}, \mathbf{D}, t)). \tag{4.6}$$

In this relation, the left-hand side of the equation represents *the approximation of the slope of* u while the right-hand represents *the slope of the approximation of* u. Since the approximations are conservative, these two sides must intersect on boxes containing a solution. Hence an empty intersection means that the boxes used in the relation do not contain the solution to the ODE system. Figure 4.1 illustrates the intuition. It is generated from an actual ordinary differential equation, considers only points instead of intervals, uses an interpolation polynomial as an approximation of u, and ignores error terms for simplicity. It illustrates how this technique can prune away a value as a potential solution at a given time. In the figure, we consider the solution to the equation that evaluates to u_0 and u_1 at t_0 and t_1 respectively. Two possible points u_2 and u_2' are then considered as possible values at t_2. The curve marked KO describes an interpolation polynomial going through u_0, u_1, u_2' at times t_0, t_1, t_2. To determine if u_2' is the value of the solution at time t_2, the idea is to test if the equation is satisfied at time t_e. (We will say more about how to choose t_e later in this work). As can be seen easily, the slope of the interpolation polynomial is different from the slope specified by f at time t_e and hence u_2' cannot be the value of the solution at t_2. The curve marked OK describes an interpolation polynomial going through u_0, u_1, u_2 at times t_0, t_1, t_2. In this case, the equation is satisfied at time t_e, which means that u_2 cannot be pruned away. The filter proposed earlier generalizes this intuition to boxes. Both the left- and the right-hand sides represent sets of slopes and the filter fails when their intersection is empty. Traditional consistency techniques and algorithms based on this filter can now be applied. For instance, one may be interested in updating the last box computed by the predictor process using the operator $D_k^* = \Box\{r \in D_k^- \mid FL(\mathbf{t}, (D_0^*, \ldots, D_{k-1}^*, r), t_e)\}$. *Observe that this operator uses an evaluation time t_e and one of the main results of this work consists in showing that t_e can be chosen optimally (in an asymptotic sense, see Chapter 7) to maximize pruning.* The following definition is a novel notion of consistency for ODEs to capture pruning of the last r boxes.

Definition 7 (Backward Consistency of Multistep Filters) *A multistep filter FL is* backward-consistent *in $(\mathbf{t}, \mathbf{D})$ for time e if*

$$D_k = \Box \left\{ u_k \in D_k \mid \exists \mathbf{u}_0 \in \mathbf{D}_0 : FL(\mathbf{t}, \mathbf{u}, e) \right\}. \tag{4.7}$$

A system of r successive multistep filters $\{FL_i\}_{0 \le i < r}$ is backward(r)-consistent *in $(\mathbf{t}_{0..k+r-1}, \mathbf{D}_{0..k+r-1})$ for time vector $(e_0, \ldots, e_{r-1})$ if*

$$\begin{aligned} \mathbf{D}_{k..k+r-1} \quad = \quad & \Box\{\mathbf{u}_{k..k+r-1} \in \mathbf{D}_{k..k+r-1} \mid \exists \mathbf{u}_0 \in \mathbf{D}_0 : \\ & \forall\ 0 \le i < r : FL_i(\mathbf{t}_{i..k+i}, \mathbf{u}_{i..k+i}, e_i)\}. \end{aligned} \tag{4.8}$$

Informally speaking, the parameter r in the definition determines the strength of the consistency, i.e., the number of backward variables on which each variable is dependent. The following proposition is an immediate consequence of Definition 7. It states that the strength of the consistency increases with the parameter r.

Proposition 2 (Property of Backward Consistency) *If a system of* $r+1$ $(r>0)$ *successive multistep filters* $\{FL_i\}_{0\leq i\leq r}$ *is backward*$(r+1)$*-consistent in* $(\mathbf{t}_{0..k+r}, \mathbf{D}_{0..k+r})$ *for time vector* $(e_0,\ldots,e_r)$*, then the system*

1. $\{FL_i\}_{0\leq i<r}$ *is backward*(r)*-consistent in* $(\mathbf{t}_{0..k+r-1}, \mathbf{D}_{0..k+r-1})$ *for time vector* $(e_0,\ldots,e_{r-1})$*;*

2. $\{FL_i\}_{1\leq i\leq r}$ *is backward*(r)*-consistent in* $(\mathbf{t}_{1..k+r}, \mathbf{D}_{1..k+r})$ *for time vector* $(e_1,\ldots,e_r)$*.*

In the next chapter, we introduce coordinate transformations in multistep filters to represent the sets of solutions compactly, i.e., to handle the wrapping effect (see Section 5.5). It is thus useful to generalize the above definition by introducing affine transformations.

Definition 8 (Generalized Backward Consistency) *Let* $Y_i \in \mathbb{IR}^n$ $(i \in \mathbb{N})$*. A multistep filter FL is* backward-consistent *in* $(\mathbf{t}, \mathbf{Y})$ *for time e if there exists an invertible affine transformation* $\mathbf{a} : \mathbb{R}^{n(k+1)} \to \mathbb{R}^{n(k+1)}$ *such that*

$$Y_k = \Box\{y_k \in Y_k \mid \exists \mathbf{y}_0 \in \mathbf{Y}_0 : FL(\mathbf{t}, \mathbf{a}(\mathbf{y}), e)\}. \qquad (4.9)$$

A system of r successive multistep filters $\{FL_i\}_{0\leq i<r}$ *is* backward(r)-consistent *in* $(\mathbf{t}_{0..k+r-1}, \mathbf{Y}_{0..k+r-1})$ *for time vector* $(e_0,\ldots,e_{r-1})$ *if there exists an invertible affine transformation* $\mathbf{a}_{0..k+r-1} : \mathbb{R}^{n(k+r)} \to \mathbb{R}^{n(k+r)}$ *such that*

$$\begin{aligned}\mathbf{Y}_{k..k+r-1} \quad = \quad & \Box\{\mathbf{y}_{k..k+r-1} \in \mathbf{Y}_{k..k+r-1} \mid \exists \mathbf{y}_0 \in \mathbf{Y}_0 : \\ & \forall 0 \leq i < r : FL_i(\mathbf{t}_{i..k+i}, \mathbf{a}_{i..k+i}(\mathbf{y}_{0..k+r-1}), e_i)\}. \qquad (4.10)\end{aligned}$$

Note that Proposition 2 also holds for generalized backward consistency. In the rest of this work, we use "backward consistency" instead of "generalized backward consistency" for simplicity. The algorithm used in our computational results enforces backward(k)-consistency of a system of k filters we describe in the next chapter.

Chapter 5

Multistep Filters

Filters rely on interval extensions of the multistep solution and of its derivative wrt t. These extensions are, in general, based on decomposing the (unknown) multistep solution into the sum of a computable approximation p and an (unknown) error term e, i.e.,

$$ms(\mathbf{t}, \mathbf{u}, t) = p(\mathbf{t}, \mathbf{u}, t) + e(\mathbf{t}, \mathbf{u}, t). \tag{5.1}$$

There exist standard techniques to build p and $\frac{\partial p}{\partial t}$ and to bound e and $\frac{\partial e}{\partial t}$. Chapter 6 reviews how they can be derived from Hermite interpolation polynomials. Here we simply assume that they are available and we show how to use them to build filters.

5.1 Natural Filters

Chapter 4 explained how natural multistep filters can be obtained by simply replacing the multistep solution ms, its derivative $\frac{\partial ms}{\partial t}$ and the function f by their interval extensions MS, DMS and F to obtain

$$DMS(\mathbf{t}, \mathbf{D}, t) = F(MS(\mathbf{t}, \mathbf{D}, t)). \tag{5.2}$$

It is not easy however to enforce backward consistency on a natural filter since the variables may occur in complex nonlinear expressions. This problem is addressed by mean-value filters that we now study.

5.2 Mean-Value Filters

5.2.1 Mean-Value Forms

As mentioned in Section 2.5, mean-value forms (MVFs) play a fundamental role in interval computations and are derived from the Mean-Value theorem. They correspond to problem linearizations around a point and result in filters that are systems of linear equations with interval coefficients and whose solutions can be enclosed reasonably efficiently. Mean-value forms are effective when the sizes of the boxes are sufficiently small, which is the case in ODEs. In addition, being linear equations, they allow for an easier treatment of the wrapping effect, a crucial problem in interval methods for ODEs presented in Section 2.4 and to be discussed in Sections 5.3 and 5.5. As a consequence, mean-value forms are especially appropriate in our context and will produce filters which are efficiently amenable to backward consistency. The rest of this section describes how to obtain mean-value filters.

5.2.2 Implicit Mean-Value Filters

Consider the function

$$\delta(\mathbf{t},\mathbf{u},e,de,t) = \frac{\partial p}{\partial t}(\mathbf{t},\mathbf{u},t) + de - f(p(\mathbf{t},\mathbf{u},t)+e). \tag{5.3}$$

If the multistep solution ms is defined at $(\mathbf{t},\mathbf{u})$, i.e. the ODE has a solution going through $u_0,\ldots,u_k$ at $t_0,\ldots,t_k$, then, by (5.1), we have the relation

$$\delta(\mathbf{t},\mathbf{u},e(\mathbf{t},\mathbf{u},t),\frac{\partial e}{\partial t}(\mathbf{t},\mathbf{u},t),t) = 0. \tag{5.4}$$

Let $\mathbf{u}^*,\mathbf{u} \in \mathbf{D}^0 \in \mathbb{IR}^{n(k+1)}$, $e^*,e \in E \in \mathbb{IR}^n$ and $de^*,de \in DE \in \mathbb{IR}^n$. By the Mean-Value theorem, we can write $(1 \le i \le n)$

$$\begin{array}{ll}\delta_i(\mathbf{t},\mathbf{u},e,de,t) & = \delta_i(\mathbf{t},\mathbf{u}^*,e^*,de^*,t) \\ & +\mathcal{J}_{(\mathbf{u},e,de)}\delta_i(\mathbf{t},\mu_i,\xi_i,\zeta_i,t)\ (\mathbf{u}-\mathbf{u}^*,e-e^*,de-de^*) \\ & = \delta_i(\mathbf{t},\mathbf{u}^*,e^*,de^*,t) + \phi_i(\mathbf{t},\mu_i,\xi_i,t)(\mathbf{u}-\mathbf{u}^*) \\ & +\psi_i(\mathbf{t},\mu_i,\xi_i,t)(e^*-e) + de_i - de_i^*\end{array} \tag{5.5}$$

where

$$\begin{array}{l}\phi_i(\mathbf{t},\mu_i,\xi_i,t) = \mathcal{J}_{\mathbf{u}}\frac{\partial p_i}{\partial t}(\mathbf{t},\mu_i,t) - \mathcal{J}f_i(p(\mathbf{t},\mu_i,t)+\xi_i)\mathcal{J}_{\mathbf{u}}p(\mathbf{t},\mu_i,t) \\ \psi_i(\mathbf{t},\mu_i,\xi_i,t) = \mathcal{J}f_i(p(\mathbf{t},\mu_i,t)+\xi_i)\end{array} \tag{5.6}$$

for some $\mu_i \in \mathbf{D}^0$, $\xi_i \in E$ and $\zeta_i \in DE$. This allows us to define a new multistep filter, which we will call an *implicit mean-value filter*. Such a filter is parametrized by the initial domain $\mathbf{D}^0$ of the variable $\mathbf{u}$.

Definition 9 (Implicit Mean-Value Filter) *An* implicit mean-value filter *for ODE $u' = f(u)$ in $\mathbf{D}^0 \in \mathbb{IR}^{n(k+1)}$ is an interval relation*

$$\begin{array}{c}FL(\mathbf{t},\mathbf{D},t) \Leftrightarrow \\ \delta(\mathbf{t},\mathbf{m}^0,m_e,m_{de},t) + \Delta(\mathbf{t},\mathbf{D}^0,E(\mathbf{t},\mathbf{D}^0,t),DE(\mathbf{t},\mathbf{D}^0,t),t)\ (\mathbf{X},E_m,DE_m) = 0\end{array} \tag{5.7}$$

where

$$\begin{array}{l}\Delta \text{ is an interval extension of the function } \mathcal{J}_{(\mathbf{u},e,de)}\delta,\\ E \text{ and } DE \text{ are interval extensions resp. of } e \text{ and } \frac{\partial e}{\partial t},\\ \mathbf{D} \subseteq \mathbf{D}^0,\\ \mathbf{X} = \mathbf{D} - \mathbf{m}^0, E_m = E(\mathbf{t},\mathbf{D}^0,t) - m_e, DE_m = DE(\mathbf{t},\mathbf{D}^0,t) - m_{de},\\ \mathbf{m}^0 = m(\mathbf{D}^0), m_e = m(E(\mathbf{t},\mathbf{D}^0,t)), m_{de} = m(DE(\mathbf{t},\mathbf{D}^0,t)).\end{array} \tag{5.8}$$

Note that the Jacobians in (5.8) can be computed by means of automatic differentiation tools (see Section 2.7). The following proposition states that an implicit mean-value filter does not eliminate any solution of the ODE. It is a direct consequence of the Mean-Value theorem.

Proposition 3 *An implicit mean-value filter for ODE $\mathcal{O}$ is a multistep filter for $\mathcal{O}$.*

5.2.3 Explicit Mean-Value Filters

In general, for initial value problems, we will be interested in pruning the last predicted box D_k^-. Hence it is convenient to derive a mean-value filter which is explicit in D_k. Let $\mathbf{D}^- \in \mathbb{IR}^{n(k+1)}$ be the predicted box of variable $\mathbf{u}$ and define $\mathbf{X}$ as $\mathbf{D} - m(\mathbf{D}^-)$. An implicit mean-value filter is an interval constraint of the form

$$\Phi(t)\mathbf{X} = \Gamma(t) \tag{5.9}$$

where $\Phi(t) \in \mathbb{IR}^{n \times n(k+1)}$ and $\Gamma(t) \in \mathbb{IR}^n$. Let us apply the midpoint technique (see Point 2 of Section 2.6) on the matrix $\Phi(t)$. We can write $\Phi(t) = m(\Phi(t)) + s(\Phi(t))$ and

$$m(\Phi(t))\mathbf{X} = \Gamma(t) - s(\Phi(t))\mathbf{X}. \tag{5.10}$$

The term $s(\Phi(t))\mathbf{X}$ is normally small (of size $\mathcal{O}(\|\omega(\mathbf{D}^-)\|^2)$) and we can substitute $\mathbf{X}$ on the right side of (5.10) for $s(\mathbf{D}^-)$, since $\mathbf{X} = \mathbf{D} - m(\mathbf{D}^-)$ and we are looking for a pruned box $\mathbf{D}^* \subseteq \mathbf{D}^-$. We obtain the system

$$m(\Phi(t))\mathbf{X} = \Gamma(t) - s(\Phi(t))s(\mathbf{D}^-). \tag{5.11}$$

Equation (5.11) can now be rewritten as

$$\sum_{i=0}^{k} A_i(t)X_i = K(t) \tag{5.12}$$

where $A_i(t) \in \mathbb{R}^{n\times n}$, $i = 0,\ldots,k$ and $K(t) \in \mathbb{IR}^n$. Let us isolate the term involving X_k :

$$A_k(t)X_k = K(t) - \sum_{i=0}^{k-1} A_i(t)X_i. \tag{5.13}$$

Multiplying both sides of (5.13) by $A_k(t)^{-1}$ (recall that $A_k(t)^{-1}$ denotes an enclosure of the inverse of $A_k(t)$) gives

$$X_k = A_k(t)^{-1}K(t) - \sum_{i=0}^{k-1} \left(A_k(t)^{-1}A_i(t)\right) X_i. \tag{5.14}$$

We are now in position to define explicit mean-value filters which play a fundamental role in our approach.

Definition 10 (Explicit Mean-Value Filter) *An* explicit mean-value filter *for ODE $\mathcal{O}$ in $\mathbf{D}^0 \in \mathbb{IR}^{n(k+1)}$ is an interval relation*

$$FL(\mathbf{t}, \mathbf{D}, t) \;\Leftrightarrow\; X_k = A_k(t)^{-1}K(t) - \sum_{i=0}^{k-1} \left(A_k(t)^{-1}A_i(t)\right) X_i \tag{5.15}$$

where

$$\begin{array}{l} \mathbf{X} = \mathbf{D} - m(\mathbf{D}^0), \\ \mathbf{D} \subseteq \mathbf{D}^0, \\ (A_0(t) \cdots A_k(t)) = m(\Phi(t)) \in \mathbb{R}^{n \times n(k+1)}, \\ K(t) = \Gamma(t) - s(\Phi(t))s(\mathbf{D}^0) \in \mathbb{IR}^n, \\ \text{the relation } \Phi(t)\mathbf{X} = \Gamma(t) \text{ is an implicit mean-value filter for } \mathcal{O} \text{ in } \mathbf{D}^0. \end{array} \tag{5.16}$$

Proposition 4 *An explicit mean-value filter for ODE $\mathcal{O}$ is a multistep filter for $\mathcal{O}$.*

It is easy to use an explicit mean-value filter to prune the predicted box D_k^- at time t_k given the boxes $D_0^*,\ldots,D_{k-1}^*$ from the previous integration steps, since X_k (and thus D_k) has been isolated. The filter simply becomes

$$D_k = m(D_k^-) + A_k(t)^{-1}K(t) - \sum_{i=0}^{k-1} \left(A_k(t)^{-1}A_i(t)\right)(D_i^* - m(D_i^*)) \tag{5.17}$$

and the pruned box D_k^* at time t_k is given by

$$D_k^* = D_k \cap D_k^-. \tag{5.18}$$

It follows directly that the explicit mean-value filter is backward-consistent in $\mathbf{D}^*$.

5.3 Problems in Mean-Value Filters

Mean-value filters often produce significant pruning of the boxes computed by the predictor process. However, they suffer from two limitations: the *wrapping effect* which is inherent in interval analysis and a *variable dependency* problem induced by the use of a multistep method. We review both of these before describing how to address them.

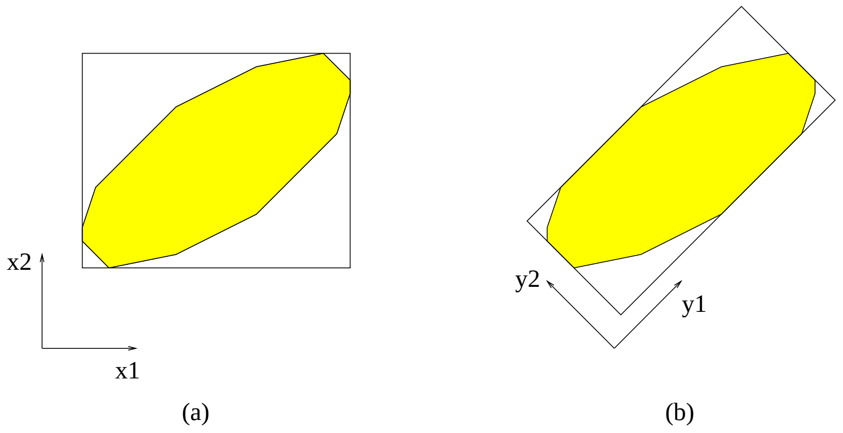

Figure 5.1: (a) A zonotope in $\mathbb{R}^2$ and the smallest enclosing box; (b) Coordinate transformation where the enclosing box fits better the zonotope.

5.3.1 Wrapping Effect

As described in Section 2.4, the wrapping effect is the name given to the over-estimation that arises from approximating a set by a box. In the context of ODEs, the set of solutions at each integration step is over-approximated by a box. These over-approximations accumulate step after step and may result in an explosion in the sizes of the computed boxes. The standard solution used in interval methods for ODEs to obtain tighter solution bounds is to choose, at each step, an appropriate local coordinate system to represent the solutions compactly (see [Loh87], [NJ99]). How does the wrapping effect occur in our context? Let us rewrite an explicit mean-value filter as

$$X_k = K(t) + \sum_{i=0}^{k-1} A_i(t) X_i \tag{5.19}$$

and let us assume that $A_0(t), \ldots, A_{k-1}(t)$ are point matrices and that $K(t)$ is a point vector. Given the boxes $X_0, \ldots, X_{k-1}$ computed at the previous steps, the exact solution set to be enclosed by X_k is

$$Z = \left\{ K(t) + \sum_{i=0}^{k-1} A_i(t) x_i \mid (x_0, \ldots, x_{k-1}) \in (X_0, \ldots, X_{k-1}) \right\}. \tag{5.20}$$

The set Z is called a *zonotope* [1] (i.e., a generalization of a parallelepiped). Figure 5.1 (a) illustrates a zonotope in $\mathbb{R}^2$ (for $k = 3$) and its smallest enclosing box. As can be seen, the box significantly overestimates the zonotope. Figure

[1]Note that W. Kühn uses zonotopes in another context, i.e. as enclosures to represent the solution sets compactly [Kuh98, Kuh98a, Kuh99].

5.1 (b) shows that the zonotope can be enclosed much more tightly by using a coordinate transformation. It should be mentioned however that finding a good coordinate system is not necessarily a trivial task (e.g., one idea is to find approximations of the main directions of the zonotope) and may not be sufficient because of the variable dependency problem that we now discuss.

5.3.2 Variable Dependencies in Explicit Filters

Consider the application of an explicit mean-value filter at two successive time steps with respective evaluation times e_0 and e_1. We obtain equations of the form:

$$\begin{array}{lcl} X_k & = & K_0(e_0) + A_{0,0}(e_0)X_0 + \ldots + A_{0,k-1}(e_0)X_{k-1}, \\ X_{k+1} & = & K_1(e_1) + A_{1,0}(e_1)X_1 + \ldots + A_{1,k-1}(e_1)X_k. \end{array} \tag{5.21}$$

The second equation computes the box X_{k+1} assuming that $X_1, \ldots, X_k$ are independent, which is not the case because of the first equation. Hence, the dependencies between $X_1, \ldots, X_k$ are lost when moving from the first to the second time step. The variable dependency problem arises because successive explicit mean-value filters overlap, i.e., each computed box X_i is used in k successive filters. One-step methods do not encounter this problem because each computed box X_i is used only at one time step to compute the following box X_{i+1}. *Global filters*, which are presented in the next section, avoid this variable dependency problem and make it possible to apply standard techniques for the wrapping effect.

5.4 Global Filters

The main idea underlying global filters is to cluster several mean-value filters together so that they do not overlap. The intuition is illustrated in Figure 5.2 for $k = 3$. It can be seen that the global filter prunes the 3 predicted boxes D_3^-, D_4^-, and D_5^- for times t_3, t_4, and t_5 using the boxes D_0^*, D_1^*, and D_2^* computed for times t_0, t_1, and t_2. Observe also that global filters do not overlap, i.e., the boxes D_0^*, D_1^*, and D_2^* will not be used in subsequent filters. More precisely, a global filter is a system of k successive explicit mean-value filters.

Definition 11 (Global Filter) *A* global filter *for ODE $\mathcal{O}$ in $\mathbf{D}^0_{0..2k-1}$ is a system $\{FL_i(\mathbf{t}_{i..k+i}, \mathbf{D}_{i..k+i}, e_i)\}_{0 \le i < k}$ of k successive explicit mean-value filters for $\mathcal{O}$ respectively in $\mathbf{D}^0_{0..k}, \ldots, \mathbf{D}^0_{k-1..2k-1}$ given as*

$$\left\{ \begin{array}{lcl} X_k & = & K_0(e_0) + A_{0,0}(e_0)X_0 + \ldots + A_{0,k-1}(e_0)X_{k-1} \\ X_{k+1} & = & K_1(e_1) + A_{1,0}(e_1)X_1 + \ldots + A_{1,k-1}(e_1)X_k \\ & \vdots & \\ X_{2k-1} & = & K_{k-1}(e_{k-1}) + A_{k-1,0}(e_{k-1})X_{k-1} + \ldots + A_{k-1,k-1}(e_{k-1})X_{2k-2} \end{array} \right. \tag{5.22}$$

where $\mathbf{X}_{0..2k-1} = \mathbf{D}_{0..2k-1} - m(\mathbf{D}^0_{0..2k-1})$.

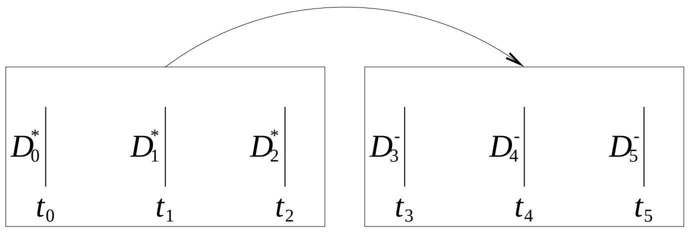

Figure 5.2: Intuition of the Globalization Process ($k = 3$).

The key idea to remove the variable dependency problem is to solve (5.22) globally by transforming the global filter into an explicit form

$$\begin{bmatrix} X_k \\ \vdots \\ X_{2k-1} \end{bmatrix} = C(\mathbf{e}_0) \begin{bmatrix} X_0 \\ \vdots \\ X_{k-1} \end{bmatrix} + R(\mathbf{e}_0) \tag{5.23}$$

or, more concisely,

$$\mathbf{X}_1 = C(\mathbf{e}_0)\mathbf{X}_0 + R(\mathbf{e}_0) \tag{5.24}$$

where $C(\mathbf{e}_0) \in \mathbb{IR}^{nk \times nk}$ and $R(\mathbf{e}_0) \in \mathbb{IR}^{nk}$.
An interesting property of global filters is that each pruned box at times t_3, t_4, or t_5 can be computed only in terms of the predicted boxes and the boxes at times t_0, t_1, and t_2 by using Gaussian elimination. Consider, for instance, a system with $k = 3$. We obtain

$$\left\{ \begin{array}{l} X_3 = A_{00}X_0 + A_{01}X_1 + A_{02}X_2 + K_0 \\ X_4 = A_{10}X_1 + A_{11}X_2 + A_{12}X_3 + K_1 \\ X_5 = A_{20}X_2 + A_{21}X_3 + A_{22}X_4 + K_2 \end{array} \right. \tag{5.25}$$

Variable X_4 can be eliminated from the last equation to obtain

$$X_5 = A_{20}X_2 + A_{21}X_3 + A_{22}(A_{10}X_1 + A_{11}X_2 + A_{12}X_3 + K_1) + K_2. \tag{5.26}$$

To avoid multiplying interval matrices (e.g., $A_{22}A_{10}$), we can apply the midpoint technique (see Point 1 of Section 2.6) to obtain

$$\begin{array}{rcl} X_5 & = & A_{20}X_2 + A_{21}X_3 + m(A_{22})(A_{10}X_1 + A_{11}X_2 + A_{12}X_3 + K_1) \\ & & +K_2 + s(A_{22})s(D_4^-). \end{array} \tag{5.27}$$

By distribution and rearrangement of the parentheses, we can rewrite (5.27) as

$$\begin{array}{rcl} X_5 & = & (m(A_{22})A_{10})X_1 + (A_{20} + m(A_{22})A_{11})X_2 + (A_{21} + m(A_{22})A_{12})X_3 \\ & & +m(A_{22})K_1 + K_2 + s(A_{22})s(D_4^-). \end{array} \tag{5.28}$$

```
function ExplicitGlobalFilter($\mathbb{O}, \mathbf{t}_0, \mathbf{D}_0^0, \mathbf{B}_{1..k-1}, \mathbf{t}_1, \mathbf{D}_1^0, \mathbf{B}_1, \mathbf{e}_0$)
  begin
    for $i := 0$ to $k-1$ do
      $\langle K_i, A_{i,0}, \ldots, A_{i,k-1} \rangle :=$ EMVFL($\mathbb{O}, \mathbf{t}_{i..i+k}, \mathbf{D}_{i..i+k}^0, \mathbf{B}_{i+1..i+k}, e_i$);
    endfor
    for $i := k-1$ downto $0$ do
      $R_i := K_i$;
      for $l := i$ downto $1$ do
        $A^* := m(A_{i,k-1})$;
        $R_i := R_i + A^* K_{l-1} + s(A_{i,k-1}) s(D_{k+l-1}^0)$;
        for $j := k-1$ downto $1$ do
          $A_{i,j} := A_{i,j-1} + A^* A_{l-1,j}$
        endfor
        $A_{i,0} := A^* A_{l-1,0}$
      endfor
    endfor
    return $\langle (A_{i,j})_{\substack{0 \le i \le k-1 \\ 0 \le j \le k-1}}, (R_i)_{0 \le i \le k-1} \rangle$
  end
```

Figure 5.3: An Algorithm for Computing an Explicit Global Filter.

Variable X_3 can be eliminated from this equation in a similar fashion to obtain a filter involving only X_5, X_0, X_1, and X_2. Similarly, variable X_3 can be eliminated from the second equation to obtain a filter only involving X_4, X_0, X_1, and X_2.

A generic algorithm for computing an explicit global filter is given in Figure 5.3. It receives as input the ODE system $\mathbb{O}$, the previous integration times $\mathbf{t}_0$, the pruned boxes $\mathbf{D}_0^0$, and the bounding boxes $\mathbf{B}_{1..k-1}$, the new integration points $\mathbf{t}_1$, the predicted boxes $\mathbf{D}_1^0$ for these integration points, the bounding boxes $\mathbf{B}_1$ for the new integration points, and the evaluation times for the filters. It generates the matrix and vectors of the explicit global filter which can be used to compute the pruned boxes. The resulting filter is backward(k)-consistent with respect to the resulting boxes. Its precise specification is as follows.

Specification 1 (ExplicitGlobalFilter) *Let B_i be a bounding box of ODE $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0), for $1 \le i \le 2k-1$. Let*

$$\langle C(\mathbf{e}_0), R(\mathbf{e}_0) \rangle = \text{ExplicitGlobalFilter}(\mathbb{O}, \mathbf{t}_0, \mathbf{D}_0^0, \mathbf{B}_{1..k-1}, \mathbf{t}_1, \mathbf{D}_1^0, \mathbf{B}_1, \mathbf{e}_0), \tag{5.29}$$

$\mathbf{X}_0 = \mathbf{D}_0 - m(\mathbf{D}_0^0)$, and $\mathbf{X}_1 = \mathbf{D}_1 - m(\mathbf{D}_1^0)$. Then, the system $\mathcal{S} : \mathbf{X}_1 = C(\mathbf{e}_0)\mathbf{X}_0 + R(\mathbf{e}_0)$ is a global filter for $\mathbb{O}$ in $(\mathbf{D}_0^0, \mathbf{D}_1^0)$.

The algorithm is generic in the sense that it uses the function EMVFL to generate an explicit mean-value filter. How to generate such a filter is discussed in Chapter 6 but its specification is given as follows.

Specification 2 (EMVFL) *Let B_i be a bounding box of ODE $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0), for $1 \leq i \leq k$. Let $\langle K(t), A_0(t), \ldots, A_{k-1}(t)\rangle =$* EMVFL$(\mathbb{O}, \mathbf{t}, \mathbf{D}^0, (B_1, \ldots, B_k), t)$. *Then, the interval relation*

$$FL(\mathbf{t}, \mathbf{D}, t) \Leftrightarrow X_k = K(t) + \sum_{i=0}^{k-1} A_i(t) X_i \tag{5.30}$$

where $\mathbf{X} = \mathbf{D} - m(\mathbf{D}^0)$ and $\mathbf{D} \subseteq \mathbf{D}^0$, is an explicit mean-value filter for $\mathbb{O}$ in $\mathbf{D}^0$.

Finally, observe that global filters not only remove the variable dependency problem by globalizing the pruning process. They also have the advantage of producing square systems which makes it possible to apply standard techniques to address the wrapping effect. The next section discusses the wrapping effect in detail.

5.5 The Wrapping Effect in Global Filters

The wrapping effect in global filters arises when multiplying a $nk \times nk$ matrix and a box of nk elements. Fortunately, since the matrices in global filters are square, the wrapping effect can be handled as in one-step methods by using local coordinate transformations and QR factorizations [Loh87]. We now explain this process in detail. Initially, starting from the previous boxes $\mathbf{D}_0^*$ and predicted boxes $\mathbf{D}_1^-$, we need to solve the system

$$\mathbf{D}_1 - m(\mathbf{D}_1^-) = C_1(\mathbf{e}_0)(\mathbf{D}_0^* - m(\mathbf{D}_0^*)) + R_1(\mathbf{e}_0) \tag{5.31}$$

or, equivalently, the system

$$\mathbf{X}_1 = C_1(\mathbf{e}_0)\mathbf{X}_0 + R_1(\mathbf{e}_0) \tag{5.32}$$

where $\mathbf{X}_1 = \mathbf{D}_1 - m(\mathbf{D}_1^-)$ and $\mathbf{X}_0 = \mathbf{D}_0^* - m(\mathbf{D}_0^*)$. The pruned boxes are then obtained by

$$\mathbf{D}_1^* = \mathbf{D}_1^- \cap (\mathbf{X}_1 + m(\mathbf{D}_1^-)). \tag{5.33}$$

The key idea in tackling the wrapping effect is to find a good coordinate system to represent the solution $\mathbf{X}_1$ compactly so that errors will not accummulate drastically in subsequent integration steps. For this purpose, we introduce a coordinate transformation

$$M_1 \mathbf{y}_1 = \mathbf{u}_1 - m(\mathbf{D}_1^*) \tag{5.34}$$

which can be reexpressed in terms of the $\mathbf{x}$ variables as

$$M_1 \mathbf{y}_1 = \mathbf{x}_1 + m(\mathbf{D}_1^-) - m(\mathbf{D}_1^*). \tag{5.35}$$

We then solve the system

$$M_1 \mathbf{Y}_1 = C_1(\mathbf{e}_0)\mathbf{X}_0 + R_1(\mathbf{e}_0) + m(\mathbf{D}_1^-) - m(\mathbf{D}_1^*) \tag{5.36}$$

by inverting the matrix M_1:

$$\mathbf{Y}_1 = (M_1^{-1} C_1(\mathbf{e}_0))\mathbf{X}_0 + M_1^{-1}(R_1(\mathbf{e}_0) + m(\mathbf{D}_1^-) - m(\mathbf{D}_1^*)). \tag{5.37}$$

The matrix M_1 and the boxes $\mathbf{Y}_1$ and $\mathbf{D}_1^*$ are then sent to the next integration step. Observe that $\mathbf{Y}_1$ is a compact representation of $\mathbf{D}_1^*$ in the local coordinate system.

In the next integration step, the boxes $\mathbf{D}_1^*$ are used (together with other data) to compute new predicted boxes $\mathbf{D}_2^-$ as well as the new global filter

$$\mathbf{D}_2 - m(\mathbf{D}_2^-) = C_2(\mathbf{e}_1)(\mathbf{D}_1^* - m(\mathbf{D}_1^*)) + R_2(\mathbf{e}_1)). \tag{5.38}$$

Since $M_1\mathbf{y}_1 = \mathbf{u}_1 - m(\mathbf{D}_1^*)$ by the coordinate transformation, the above filter can be rewritten into

$$\mathbf{X}_2 = (C_2(\mathbf{e}_1)M_1)\mathbf{Y}_1 + R_2(\mathbf{e}_1) \tag{5.39}$$

where $\mathbf{X}_2 = \mathbf{D}_2 - m(\mathbf{D}_2^-)$. Observe the associativity of the multiplication which is critical in reducing the wrapping effect. The new boxes are computed as

$$\mathbf{D}_2^* = \mathbf{D}_2^- \cap (\mathbf{X}_2 + m(\mathbf{D}_2^-)). \tag{5.40}$$

Once again, we would like to represent the set of solutions $\mathbf{X}_2$ compactly and we use a local coordinate transformation

$$M_2\mathbf{y}_2 = \mathbf{u}_2 - m(\mathbf{D}_2^*) \tag{5.41}$$

to obtain the system

$$M_2\mathbf{Y}_2 = (C_2(\mathbf{e}_1)M_1)\mathbf{Y}_1 + R_2(\mathbf{e}_1) + m(\mathbf{D}_2^-) - m(\mathbf{D}_2^*). \tag{5.42}$$

This equation system can be solved by inverting M_2:

$$\mathbf{Y}_2 = (M_2^{-1}(C_2(\mathbf{e}_1)M_1))\mathbf{Y}_1 + M_2^{-1}(R_2(\mathbf{e}_1) + m(\mathbf{D}_2^-) - m(\mathbf{D}_2^*)). \tag{5.43}$$

Once again, observe the associativity in the multiplication to tackle the wrapping effect. The hope is that the matrix $M_2^{-1}(C_2(\mathbf{e}_1)M_1)$ be diagonally dominant or triangular. Also, M_2, $\mathbf{Y}_2$, and $\mathbf{D}_2^*$ will be sent to the next integration step. As a consequence, at integration step i, we solve

$$\mathbf{X}_i = (C_i(\mathbf{e}_{i-1})M_{i-1})\mathbf{Y}_{i-1} + R_i(\mathbf{e}_{i-1}) \tag{5.44}$$

where $\mathbf{X}_i = \mathbf{D}_i - m(\mathbf{D}_i^-)$ and the new boxes are obtained by

$$\mathbf{D}_i^* = \mathbf{D}_i^- \cap (\mathbf{X}_i + m(\mathbf{D}_i^-)). \tag{5.45}$$

The local coordinate transformation

$$M_i\mathbf{y}_i = \mathbf{u}_i - m(\mathbf{D}_i^*) \tag{5.46}$$

```
function Prune($\mathbb{O}, \mathbf{t}_0, \mathbf{D}_0^*, \mathbf{B}_{1..k-1}, \mathbf{Y}_0, M_0, \mathbf{t}_1, \mathbf{D}_1^-, \mathbf{B}_1$)
  begin
  1   $\langle C_1, R_1\rangle := $ ExplicitGlobalFilter($\mathbb{O}, \mathbf{t}_0, \mathbf{D}_0^*, \mathbf{B}_{1..k-1}, \mathbf{t}_1, \mathbf{D}_1^-, \mathbf{B}_1, \mathbf{e}_0$);
  2   $C_1^* = C_1 M_0$;
  3   $\mathbf{X}_1 := C_1^*\mathbf{Y}_0 + R_1$;
  4   $\mathbf{D}_1^* := (\mathbf{X}_1 + m(\mathbf{D}_0^-)) \cap (\mathbf{D}_0^-)$;
  5   $M_1 := $ CoordTransfo($C_1^*, \mathbf{Y}_0$);
  6   $\mathbf{d}_1 := m(\mathbf{D}_1^-) - m(\mathbf{D}_1^*)$;
  7   $\mathbf{Y}_1 := (m(M_1^{-1})C_1^*)\mathbf{Y}_0 + m(M_1^{-1})(R_1 + \mathbf{d}_1) + s(M_1^{-1})(\mathbf{X}_1 + \mathbf{d}_1)$;
  8   return $\langle \mathbf{D}_1^*, \mathbf{Y}_1, M_1\rangle$
  end
```

Figure 5.4: The Pruning Algorithm on Global Filters.

is used to compute the new $\mathbf{Y}_i$ which is given by

$$\mathbf{Y}_i = (M_i^{-1}(C_i(\mathbf{e}_{i-1})M_{i-1}))\mathbf{Y}_{i-1} + M_i^{-1}(R_i(\mathbf{e}_{i-1}) + m(\mathbf{D}_i^-) - m(\mathbf{D}_i^*)). \tag{5.47}$$

In addition, in order to avoid the costly (see [Knu94]) product of the two interval matrices M_i^{-1} and $C_i(\mathbf{e}_{i-1})M_{i-1}$, we use the standard midpoint technique (see Point 1 of Section 2.6) to obtain

$$\begin{aligned}\mathbf{Y}_i \;=\;& (m(M_i^{-1})(C_i(\mathbf{e}_{i-1})M_{i-1}))\mathbf{Y}_{i-1} + m(M_i^{-1})(R_i(\mathbf{e}_{i-1}) + \mathbf{d}_i) + \\ & s(M_i^{-1})((C_i(\mathbf{e}_{i-1})M_{i-1})\mathbf{Y}_{i-1} + R_i(\mathbf{e}_{i-1}) + \mathbf{d}_i)\end{aligned} \tag{5.48}$$

where $\mathbf{d}_i = m(\mathbf{D}_i^-) - m(\mathbf{D}_i^*)$. This last system can be rewritten into

$$\begin{aligned}\mathbf{Y}_i \;=\;& (m(M_i^{-1})(C_i(\mathbf{e}_{i-1})M_{i-1}))\mathbf{Y}_{i-1} + m(M_i^{-1})(R_i(\mathbf{e}_{i-1}) + \mathbf{d}_i) \\ & + s(M_i^{-1})(\mathbf{X}_i + \mathbf{d}_i)\end{aligned} \tag{5.49}$$

by definition of $\mathbf{X}_i$. In this process, the choice of an appropriate matrix M_i is, of course, crucial. Lohner's QR factorization technique [Loh87] is a very successful scheme to obtain such a matrix.

5.6 A Pruning Algorithm based on Global Filters

We are now in position to present a pruning algorithm based on global filters. The pruning algorithm enforces backward(k)-consistency on a global filter composed of k mean-value filters. The algorithm is shown in Figure 5.4 and its specification is as follows.

Specification 3 (Prune) *Let ms be the multistep solution of ODE $\mathbb{O}$ and B_i a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0) for $1 \le i \le 2k-1$. Let*

$$\langle \mathbf{D}_1^*, \mathbf{Y}_1, M_1\rangle = \text{Prune}(\mathbb{O}, \mathbf{t}_0, \mathbf{D}_0^*, \mathbf{B}_{1..k-1}, \mathbf{Y}_0, M_0, \mathbf{t}_1, \mathbf{D}_1^-, \mathbf{B}_1), \tag{5.50}$$

$\mathcal{A}_0 = \{M_0\mathbf{y}_0 + m(\mathbf{D}_0^*) \mid \mathbf{y}_0 \in \mathbf{Y}_0\} \cap \mathbf{D}_0^*$ *and* $\mathcal{A}_1 = \{M_1\mathbf{y}_1 + m(\mathbf{D}_1^*) \mid \mathbf{y}_1 \in \mathbf{Y}_1\} \cap \mathbf{D}_1^*$. *Then,*

1. $ms((\mathbf{t}_0,\mathbf{t}_1),(\mathcal{A}_0,\mathbf{D}_1^-),t_i) \subseteq ms((\mathbf{t}_0,\mathbf{t}_1),(\mathcal{A}_0,\mathcal{A}_1),t_i)$, *for* $k \leq i \leq 2k-1$*;*
2. $\mathbf{D}_1^* \subseteq \mathbf{D}_1^-$*;*
3. *there exists a global filter which is backward(k)-consistent in* $((\mathbf{t}_0,\mathbf{t}_1),(\mathbf{Y}_0,\mathbf{D}_1^*))$ *and in* $((\mathbf{t}_0,\mathbf{t}_1),(\mathbf{Y}_0,\mathbf{Y}_1))$ *for a given time vector.*

The algorithm receives as input the ODE $\mathcal{O}$, the previous integration times $\mathbf{t}_0$, the pruned boxes $\mathbf{D}_0^*$ computed at times $\mathbf{t}_0$, the bounding boxes $\mathbf{B}_{1..k-1}$ for all previous integration steps, the boxes $\mathbf{Y}_0$ and matrix M_0 from the previous integration step as well as the new integration times $\mathbf{t}_1$, the predicted boxes $\mathbf{D}_1^-$ and the bounding boxes $\mathbf{B}_1$ for these integration times. It returns the pruned boxes $\mathbf{D}_1^*$ for integration steps $\mathbf{t}_1$ as well as the new boxes $\mathbf{Y}_1$ and the new matrix M_1 to be used in the next integration step. The algorithm itself follows the same steps as outlined in the preceeding section. It computes the explicit form of the global filter (line 1), the new boxes $\mathbf{X}_1$ (line 2), and the pruned boxes $\mathbf{D}_1^*$ (line 3). It then computes the new matrix M_1 (line 4) and the new boxes $\mathbf{Y}_1$ (line 6).

Chapter 6

Hermite Filters

In the previous chapter, we assumed the existence of interval extensions of p and $\partial p/\partial t$ and we assumed that we could bound the error terms e and $\partial e/\partial t$. We now show how to use Hermite interpolation polynomials for this purpose.

6.1 Definition

Informally speaking, a Hermite interpolation polynomial approximates a function $g \in C^r$ (for sufficiently large r) which is known implicitly by its values and the values of its successive derivatives at various points. A Hermite interpolation polynomial is specified by imposing that its values and the values of its successive derivatives at some given points be equal to the values of g and of its derivatives at the same points. Note that the number of conditions (i.e., the number of successive derivatives that are considered) may vary at the different points [SB80, Atk88, KC96].

Definition 12 (Hermite(σ) Interpolation Polynomial) *Consider the ODE $u' = f(u)$ and let $\sigma = (\sigma_0, \ldots, \sigma_k) \in \mathbb{N}^{k+1}$, $\sigma_i \neq 0$ $(0 \leq i \leq k)$ and $\sigma_s = \sum_{i=0}^{k} \sigma_i$. The* Hermite($\sigma$) interpolation polynomial *wrt f and $(\mathbf{t}, \mathbf{u})$ is the unique polynomial q of degree $\leq \sigma_s - 1$ satisfying*

$$q^{(j)}(t_i) = j!(u_i)_j \quad (0 \leq j \leq \sigma_i - 1,\ 0 \leq i \leq k). \tag{6.1}$$

Proposition 5 (Hermite(σ) Interpolation Polynomial) *The polynomial q satisfying the conditions (6.1) is given by*

$$q(t) = \sum_{i=0}^{k} \sum_{j=0}^{\sigma_i - 1} j!(u_i)_j \varphi_{ij}(t) \tag{6.2}$$

where

$$\begin{array}{rcl}\varphi_{i,\sigma_i-1}(t) & = & l_{i,\sigma_i-1}(t), \quad i=0,\ldots,k,\\ \varphi_{ij}(t) & = & l_{ij}(t) - \sum_{\nu=j+1}^{\sigma_i-1} l_{ij}^{(\nu)}(t_i)\varphi_{i\nu}(t), \quad i=0,\ldots,k, \ j=0,\ldots,\sigma_i-2,\\ l_{ij}(t) & = & \frac{(t-t_i)^j}{j!}\prod_{\substack{\nu=0\\ \nu\neq i}}^{k}\left(\frac{t-t_\nu}{t_i-t_\nu}\right)^{\sigma_\nu}, \quad i=0,\ldots,k, \ j=0,\ldots,\sigma_i-1.\end{array} \tag{6.3}$$

It is easy to take interval extensions of a Hermite interpolation polynomial and of its derivative. The Taylor coefficients $(D_i)_j$ of the solution specifying the derivative conditions at the various interpolation points, as well as their Jacobians $\mathcal{J}(D_i)_j$ needed in the mean-value Hermite filters, can be computed by automatic differentiation techniques (see Section 2.7).

6.2 Bounding the Error Terms

The only remaining issue is to bound the error terms. The following standard theorem (e.g., [SB80, Atk88, KC96]) provides the necessary theoretical basis.

Theorem 3 (Hermite Error Term) *Let $p(\mathbf{t},\mathbf{u},t)$ be the Hermite(σ) interpolation polynomial in t wrt f and $(\mathbf{t},\mathbf{u})$. Let $u(t) = ms(\mathbf{t},\mathbf{u},t)$, $ms(\mathbf{t},\mathbf{u},t) = p(\mathbf{t},\mathbf{u},t) + e(\mathbf{t},\mathbf{u},t)$, $T = \Box\{t_0,\ldots,t_k,t\}$, $\sigma_s = \sum_{i=0}^{k}\sigma_i$ and $w(t) = \prod_{i=0}^{k}(t-t_i)^{\sigma_i}$. We have $(1 \le i \le n)$*

1. *$\exists\ \xi_i \in T : e_i(\mathbf{t},\mathbf{u},t) = \frac{1}{\sigma_s!}u_i^{(\sigma_s)}(\xi_i)w(t)$;*
2. *$\exists\ \xi_{1,i},\xi_{2,i} \in T : \frac{\partial e_i}{\partial t}(\mathbf{t},\mathbf{u},t) = \frac{1}{\sigma_s!}u_i^{(\sigma_s)}(\xi_{1,i})w'(t) + \frac{1}{(\sigma_s+1)!}u_i^{(\sigma_s+1)}(\xi_{2,i})w(t)$.*

How to use this theorem to bound the error terms? If B is a bounding box (produced by the bounding box process - see Section 3.1) for the ODE over $T = \Box\{t_0,\ldots,t_k,t\}$ wrt $(\mathbf{t}_0,\mathbf{u}_0)$, it suffices to compute two boxes $(B)_{\sigma_s}$ and $(B)_{\sigma_s+1}$ by automatic differentiation (see Section 2.7). We then obtain

$$\begin{array}{rcl} e(\mathbf{t},\mathbf{u},t) & \in & (B)_{\sigma_s}w(t);\\ \frac{\partial e}{\partial t}(\mathbf{t},\mathbf{u},t) & \in & (B)_{\sigma_s}w'(t) + (B)_{\sigma_s+1}w(t).\end{array} \tag{6.4}$$

As a consequence, we can compute an effective relaxation of the ODE by specializing global filters with a Hermite interpolation polynomial and its error bound. In the following, filters based on Hermite(σ) interpolation are called *Hermite(σ) filters* and a global Hermite(σ) filter is denoted by GHF(σ). Note that Appendix D discusses how to evaluate Hermite polynomials accurately.

Chapter 7

Optimal Hermite Filters

Let us summarize what we have achieved so far. The basic idea of our approach is to approximate the ODE $\forall\ t :\ u' = f(u)$ by a filter

$$\forall\ t : FL(\mathbf{t}, \mathbf{D}, t). \tag{7.1}$$

We have shown that a global filter which prunes the last k boxes by using k successive mean-value filters addresses the wrapping effect and the variable dependency problem. We have also shown that a global filter can be obtained by using Hermite interpolation polynomials together with their error bounds. As a consequence, we obtain a filter

$$\forall\ \mathbf{e}_0 : GHF(\sigma)(\mathbf{t}, \mathbf{D}, \mathbf{e}_0) \tag{7.2}$$

which can be used to prune the last k predicted boxes. The main remaining issue is to find an *evaluation time vector* $\mathbf{e}_0$ which miminizes the sizes of the solution boxes in

$$GHF(\sigma)(\mathbf{t}, \mathbf{D}, \mathbf{e}_0). \tag{7.3}$$

The purpose of this chapter is to show that there exists an optimal evaluation time vector (in a precise sense that we will define) and that it can be approximated or computed efficiently.

7.1 Preview of the Approach

Our goal is to find an evaluation time vector $\mathbf{e}_0$ which miminizes the sizes of the solution boxes in a global Hermite filter. However, this is a difficult problem in general. We will thus solve a simpler problem, which consists in choosing an evaluation time that minimizes the *local error* of an individual filter, i.e., the size of the enclosure of $ms(\mathbf{t}_0, \mathbf{u}_0, t_k)$ produced by the filter, assuming that the *point* values $u_0, \ldots, u_{k-1}$ are given (and, of course, that $ms(\mathbf{t}_0, \mathbf{u}_0, t_k)$ is defined).

Definition 13 (Local Error of a Filter) *Let FL be a filter for ODE $u' = f(u)$. The local error of FL wrt $(\mathbf{t}_0, \mathbf{u}_0, t)$, denoted by $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t)$, is defined as*

$$e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \omega\left(\Box\{u_k \in \mathbb{R}^n \mid FL(\mathbf{t}, \mathbf{u}, t)\}\right). \tag{7.4}$$

Since in a global filter we compute k boxes in one step, the step size is defined as $h = t_k - t_0$. Our analysis is based on the assumption that the step size h is sufficiently small. When we talk about an optimal evaluation time, the term *optimal* is thus to be understood in an *asymptotic* sense.

In the following, we restrict our attention to Hermite filters which satisfy a certain hypothesis (Section 7.2). To find an optimal evaluation time, we first derive the local error (Section 7.3). From the local error, we can then characterize the optimal evaluation time (Section 7.4). Two of the main results of this chapter are as follows:

1. For a sufficiently small step size h, the relative distance $(t_e - t_k)/h$ between the optimal evaluation time t_e and the point t_k in a Hermite(σ) filter depends only on the relative distances $(t_{i+1} - t_i)/h$ $(i = 0, \ldots, k-1)$ between the interpolation points $t_0, \ldots, t_k$ and on σ. In particular, it does not depend on the ODE itself;

2. From a practical standpoint, the computation of the optimal evaluation time induces a negligible overhead of the method. In particular, if we assume $t_{i+1} - t_i = h/k$ $(i \in \mathbb{N})$, the relative distance between the optimal evaluation time and t_k can be precomputed once for all for given k and σ.

The third main result is concerned with the order of a Hermite($(\sigma_0, \ldots, \sigma_k)$) filter which is shown to be $\mathcal{O}(h^{\sigma_s+1})$ where $\sigma_s = \sum_{i=0}^{k} \sigma_i$ when the evaluation point is chosen carefully.

7.2 Assumptions and Notations

The following assumptions are used in this chapter. We assume that the integration times are increasing, i.e. $t_0 < \ldots < t_k$, and that $t - t_k = \mathcal{O}(h)$. We also assume that the function f satisfies a Lipschitz condition on $\Omega \subseteq \mathbb{R}^n$:

$$\exists c \in \mathbb{R}, \forall u, v \in \Omega : \|f(u) - f(v)\| \leq c\|u - v\|. \tag{7.5}$$

Note that (7.5) holds if we assume $f \in C^1(\Omega)$. We further assume that the interval extension F of function f satisfies $(D \subseteq \Omega)$

$$\omega(F(D)) = \mathcal{O}(\omega(D)). \tag{7.6}$$

For instance, (7.6) holds if F is the natural interval extension of f (see Section 2.3) and (7.5) holds. We also assume that B is a bounding box of $u' = f(u)$ over $T = \Box\{t_0, \ldots, t_k, t\}$ wrt $(\mathbf{t}_0, \mathbf{u}_0)$ and that (see [Ned99])

$$\omega\left((B)_j\right) = \Theta(\omega(B)) = \Theta(h), \quad j \in \mathbb{N}. \tag{7.7}$$

From (7.5), the condition (7.7) holds if $(B)_j$ is a sufficiently tight enclosure of the set $\{(x)_j \mid x \in B\}$ (this can be obtained by using the natural interval extension of the recursive relation (2.63) in Section 2.7). In addition, we assume that the multistep solution *ms* is defined at $(\mathbf{t}_0, \mathbf{u}_0)$ or, in other words, that the ODE has a solution going through $u_0, \ldots, u_{k-1}$ at times $t_0, \ldots, t_{k-1}$. We also use the notations $\sigma = (\sigma_0, \ldots, \sigma_k)$, $\sigma_s = \sum_{i=0}^{k} \sigma_i$, and $w(t) = \prod_{i=0}^{k}(t - t_i)^{\sigma_i}$.

Since we are interested in computing an enclosure of $ms(\mathbf{t}_0, \mathbf{u}_0, t_k)$ from the *point* values $u_0, \ldots, u_{k-1}$, we will consider a Hermite filter *FL* satisfying

$$FL(\mathbf{t}, (\mathbf{u}_0, v), t) \Rightarrow \frac{\partial p}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) + DE(t) - F(p(\mathbf{t}, (\mathbf{u}_0, v), t) + E(t)) = 0 \quad (7.8)$$

where

- F is an interval extension of f;
- $E(t) = (B)_{\sigma_s} w(t)$;
- $DE(t) = (B)_{\sigma_s} w'(t) + (B)_{\sigma_s+1} w(t)$;
- $p(\mathbf{t}, (\mathbf{u}_0, v), t)$ is the Hermite(σ) interpolation polynomial in t wrt f and $(\mathbf{t}, (\mathbf{u}_0, v))$.

Let us introduce the function

$$\delta(\mathbf{t}, (\mathbf{u}_0, v), t) = \frac{\partial p}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) - f(p(\mathbf{t}, (\mathbf{u}_0, v), t) + m_e(t)) \quad (7.9)$$

where $m_e(t) = m(E(t))$. From the hypothesis (7.6), the condition (7.8) can be rewritten as

$$FL(\mathbf{t}, (\mathbf{u}_0, v), t) \Rightarrow \delta(\mathbf{t}, (\mathbf{u}_0, v), t) = -DE(t) + \mathcal{O}(\omega(E(t))). \quad (7.10)$$

In the case (7.6), the condition (7.10) is satisfied for natural Hermite filters (see Section 5.1), provided that the interval extensions *MS* and *DMS* of *ms* and $\frac{\partial ms}{\partial t}$ yield point values when evaluated at point arguments (recall that we assume exact interval arithmetic for the theoretical parts of this work). If we assume that the interval extension of the Jacobian of f satisifies the same condition as F, i.e. $\omega(\mathcal{J}(D)_0) = \mathcal{O}(\omega(D))$, then (7.10) is satisfied for *implicit mean-value* Hermite filters, and well approximated for *explicit mean-value* Hermite filters if the matrix inversion is accurate (see Section 5.2).

Let us denote the Jacobian of δ wrt variable v as follows:

$$\begin{aligned} \Phi(t, v) &= \mathcal{J}_v \delta(\mathbf{t}, (\mathbf{u}_0, v), t) \\ &= \mathcal{J}_v \tfrac{\partial p}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) - \mathcal{J} f(p(\mathbf{t}, (\mathbf{u}_0, v), t) + m_e(t)) \mathcal{J}_v p(\mathbf{t}, (\mathbf{u}_0, v), t). \end{aligned} \quad (7.11)$$

Finally, we introduce the following functions:

$$\begin{array}{l}
\lambda(t) = \left(\left(\sum_{j=0}^{\sigma_k-2} \beta_{j+1} \frac{(t-t_k)^j}{j!}\right) + \left(\sum_{j=0}^{\sigma_k-1} \beta_j \frac{(t-t_k)^j}{j!}\right) \sum_{\nu=0}^{k-1} \frac{\sigma_\nu}{t-t_\nu}\right) \pi(t); \\
\beta_0 = 1, \beta_j = -\pi^{(j)}(t_k),\ j = 1, \ldots, \sigma_k - 1; \\
\pi(t) = \prod_{\nu=0}^{k-1} \left(\frac{t-t_\nu}{t_k-t_\nu}\right)^{\sigma_\nu}; \\
\\
\gamma(t) = \sum_{i=0}^{k} \frac{\sigma_i}{t-t_i}.
\end{array} \tag{7.12}$$

7.3 Local Error of a Hermite Filter

To characterize the local error of a Hermite filter, we first need a technical lemma which characterizes the behavior of the derivatives of the filter.

Lemma 1 *We have*

1. $\Phi(t, v) = I\lambda(t) + \mathcal{O}(1)$;
2. $\lambda(t) = \mathcal{O}(h^{-1})$;
3. $\lambda(t) = \Theta(h^{-1})$ *for* $t_{k-1} < t < t_k$.

This lemma shows that $\Phi(t, v)$ is a $\Theta(h^{-1})$ asymptotically diagonal matrix for $t_{k-1} < t < t_k$. Its proof is given in Appendix C. We are now in position to characterize the local error of a Hermite filter.

Theorem 4 (Local Error of a Hermite Filter) *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying (7.10). We have*

1. $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = |(I\lambda(t) + \mathcal{O}(1))^{-1}|\Theta(\omega(B))\,(|w'(t)| + |w(t)|)$;
2. $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \Omega(h^2)\,(|w'(t)| + |w(t)|)$;
3. *If* $t_{k-1} < t < t_k$, *then* $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \Theta(h^2)\,(|w'(t)| + |w(t)|)$.

Proof Consider two arbitrary vectors $v_1, v_2 \in \mathbb{R}^n$ such that

$$FL(\mathbf{t}, (\mathbf{u}_0, v_1), t) \quad \& \quad FL(\mathbf{t}, (\mathbf{u}_0, v_2), t). \tag{7.13}$$

By the Mean-Value theorem, we can write

$$\delta(\mathbf{t}, (\mathbf{u}_0, v_2), t) - \delta(\mathbf{t}, (\mathbf{u}_0, v_1), t) = \Phi(t, \nu)(v_2 - v_1) \tag{7.14}$$

where ν is on the straight line between v_1 and v_2. When the matrix $\Phi(t, \nu)$ is regular, we can write by Lemma 1 and (7.10)

$$\begin{array}{rcl}
v_2 - v_1 & = & \Phi^{-1}(t, \nu)\,(\delta(\mathbf{t}, (\mathbf{u}_0, v_2), t) - \delta(\mathbf{t}, (\mathbf{u}_0, v_1), t)) \\
& = & (I\lambda(t) + \mathcal{O}(1))^{-1}\,(DE(t) - DE(t) + \mathcal{O}(\omega(E(t)))).
\end{array} \tag{7.15}$$

Since the two vectors v_1 and v_2 are chosen arbitrarily, it follows from (7.7) that

$$\begin{aligned} e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) &= |(I\lambda(t) + \mathcal{O}(1))^{-1}| \left(\omega(DE(t)) + \mathcal{O}(\omega(E(t)))\right) \\ &= |(I\lambda(t) + \mathcal{O}(1))^{-1}| \Theta(\omega(B)) \left(|w'(t)| + |w(t)|\right) \end{aligned} \tag{7.16}$$

which proves Point 1. Points 2 and 3 are now direct consequences of Lemma 1 and (7.7). □

We are now ready to show how to find an optimal evaluation time for Hermite filters.

7.4 Optimal Evaluation Time for a Hermite Filter

Our first result is fundamental and characterizes the order of a Hermite filter. It also hints on how to obtain an optimal evaluation time. Recall that the order of a method (or of a filter) is the order of the local error minus 1.

Theorem 5 (Order of a Hermite Filter) *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying (7.10). Then,*

1. *There exists t such that $t_{k-1} < t < t_k$ and $w'(t) = 0$;*
2. *If $t_{k-1} < t < t_k$ and $w'(t) = 0$, then $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \mathcal{O}(h^{\sigma_s+2})$;*
3. *If $w'(t) \neq 0$, then $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \Omega(h^{\sigma_s+1})$.*

Proof Consider an evaluation time t such that $t - t_k = \mathcal{O}(h)$. We have $w(t) = \mathcal{O}(h^{\sigma_s})$ and $w'(t) = \mathcal{O}(h^{\sigma_s - 1})$. First assume that $t_{k-1} < t < t_k$ and $w'(t) = 0$. By Rolle's theorem, since $w(t_{k-1}) = w(t_k) = 0$, there exists such an evaluation time t. By Theorem 4, $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \mathcal{O}(h^{\sigma_s+2})$. Now assume that $w'(t) \neq 0$. By Theorem 4, $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \Omega(h^{\sigma_s+1})$. □

Observe that the above theorem indicates that a better order for Hermite filters is obtained when we choose an evaluation time t that is a root of the polynomial w'. This is the basis of our next result which describes a necessary condition for optimality.

Theorem 6 (Necessary Condition for Optimal Hermite Filters) *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying (7.10) and let $t_e \in \mathbb{R}$ be such that*

$$e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t_e) = \min_{t - t_k = \mathcal{O}(h)} \{e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t)\} \tag{7.17}$$

for h sufficiently small. Then, t_e is a zero of the function γ.

Proof Assume that $t - t_k = \mathcal{O}(h)$ and that h is sufficiently small. By Theorem 5, $w'(t_e)$ must be zero to minimize the local error. Note that $FL(\mathbf{t}, (\mathbf{u}_0, v), t_i)$ holds for *any* $v \in \mathbb{R}^n$ if $w'(t_i) = 0$ $(0 \le i \le k)$. Thus $t_e \notin \{t_0, \ldots, t_k\}$ and $w(t_e) \neq 0$. Since $w'(t) = w(t)\gamma(t)$, we conclude that $\gamma(t_e) = 0$. □

Our next result specifies the number of zeros of the function γ as well as their locations.

Proposition 6 *The function γ in Theorem 6 has exactly k zeros $s_0, \ldots, s_{k-1}$ such that $t_i < s_i < t_{i+1}$ $(0 \le i < k)$.*

Proof We have $w'(t) = w(t)\gamma(t)$. By Rolle's theorem, as $w(t_i) = w(t_{i+1}) = 0$, w' has a root s_i with $t_i < s_i < t_{i+1}$ and $w(s_i) \neq 0$ $(0 \le i < k)$. Furthermore, the roots of w' are in $\{s_0, \ldots, s_{k-1}, t_0, \ldots, t_k\}$ because t_i is a root of multiplicity $\sigma_i - 1$ $(0 \le i \le k)$ and w' is of degree $\sigma_s - 1$, i.e., $k + \sum_{i=0}^{k}(\sigma_i - 1) = \sigma_s - 1$. Since γ is not defined at $t_0, \ldots, t_k$, its zeros are in $\{s_0, \ldots, s_{k-1}\}$. □

We are now ready to characterize precisely the optimal evaluation time for a Hermite filter.

Theorem 7 (Optimal Evaluation Time) *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying (7.10), let $s_0 < \ldots < s_{k-1}$ be the zeros of γ, and let $t_e \in \mathbb{R}$ such that*

$$e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t_e) = \min_{t - t_k = \mathcal{O}(h)} \{e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t)\}. \tag{7.18}$$

Then, for h sufficiently small,

$$|(w/\lambda)(t_e)| = \min_{t \in \{s_0, \ldots, s_{k-1}\}} \{|(w/\lambda)(s)|\}. \tag{7.19}$$

Proof Let us assume that h is sufficiently small. From Theorem 6, we know that $t_e \in \{s_0, \ldots, s_{k-1}\}$. By definition, for $i = 0, \ldots, k-1$, $w'(s_i) = w(s_i)\gamma(s_i) = 0$ and, from Theorem 4,

$$e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, s_i) = |(I\lambda(s_i) + \mathcal{O}(1))^{-1}|\Theta(\omega(B))|w(s_i)|. \tag{7.20}$$

From Proposition 6, if $t = s_i$ $(i = 0, \ldots, k-1)$, B is a bounding box over $T = \Box\{t_0, \ldots, t_k, s_i\} = [t_0, t_k]$ wrt $(\mathbf{t}_0, \mathbf{u}_0)$ and the factor $\Theta(\omega(B))$ does not depend on $t = s_i$. We have thus to minimize the function

$$\rho(t) = |(I\lambda(t) + \mathcal{O}(1))^{-1}||w(t)| \tag{7.21}$$

for $t \in \{s_0, \ldots, s_{k-1}\}$. By Lemma 1, $\lambda(s_{k-1}) = \Theta(h^{-1})$. Therefore, we must have $\lambda(t_e) = \Theta(h^{-1})$ and $\rho(t_e) \approx |(w/\lambda)(t_e)|$. Let us now assume that there exists $i \in 0..k-1$ such that $|(w/\lambda)(s_i)| < |(w/\lambda)(t_e)|$. We can write

$$\begin{aligned} |(w/\lambda)(s_i)| < |(w/\lambda)(t_e)| \quad &\Rightarrow \quad \lambda(s_i) = \Theta(h^{-1}) \\ &\Rightarrow \quad \rho(s_i) \approx |(w/\lambda)(s_i)| \\ &\Rightarrow \quad \rho(s_i) < \rho(t_e) \end{aligned} \tag{7.22}$$

which is a contradiction. □

k	1	2	3	4	5	6
$(t_e - t_k)/h$	-0.5000	-0.2113	-0.1273	-0.0889	-0.0673	-0.0537

Table 7.1: Relative Distance between the Rightmost Zero t_e of γ and t_k when $\sigma_0 = \ldots = \sigma_k$.

7.5 Discussion

It is important to discuss the consequences of Theorem 6 in some detail. First observe that the *relative distance* $(t_e - t_k)/h$ between the optimal evaluation time t_e and the point t_k depends *only* on the *relative distances* $(t_{i+1} - t_i)/h$ $(i = 0, \ldots, k-1)$ between the interpolation points $t_0, \ldots, t_k$ and on the vector σ. In particular, it is independent from the ODE itself. For instance, for $k = 1$, we have $\gamma(t) = \frac{\sigma_0}{t-t_0} + \frac{\sigma_1}{t-t_1}$ and γ has a single zero given by $t_e = \frac{\sigma_1 t_0 + \sigma_0 t_1}{\sigma_0 + \sigma_1}$. In addition, if $\sigma_0 = \ldots = \sigma_k$, then the zeros of γ are independent from σ. In particular, for $k = 1$, we have $t_e = (t_0 + t_1)/2$. From a practical standpoint, the computation of the optimal evaluation time induces a negligible overhead of the method. In particular, if we assume $t_{i+1} - t_i = h/k$ $(i \in \mathbb{N})$, then the relative distance between t_k and the optimal evaluation time can be precomputed and stored for a variety of values of k and σ. Finally, it is worth stressing that any zero of function γ gives an $\mathcal{O}(h^{\sigma_s+1})$ order for the Hermite filter provided that $\lambda(t) = \Theta(h^{-1})$ at that zero. Hence any such zero is in fact a potential candidate for the optimal evaluation time. In our experiments (see the next section), the right-most zero was always the optimal evaluation time when $\sigma_0 = \ldots = \sigma_k$, although we have not been able to prove this result.

7.6 Illustration

We now illustrate the theoretical results presented in this chapter. Table 7.1 gives approximative values of the relative distance $(t_e - t_k)/h$ between the right-most zero t_e of the function γ and the point t_k $(1 \leq k \leq 6)$, for $\sigma_0 = \ldots = \sigma_k$ and $t_{i+1} - t_i = h/k$ $(i = 0, \ldots, k-1)$. Observe that, for two interpolation points, t_e is in the middle of t_0 and t_1. It then moves closer and closer to t_k for larger values of k.

Figures 7.1, 7.2, 7.3, 7.4, 7.5, 7.6 and 7.7 illustrate the functions γ, w, w', λ and w/λ, and their sometimes complex interactions, for several values of k and σ. In each case, the top-left figure shows the function w' and γ, as well as the zeros of γ. The top-right figure shows the function w with the zeros of γ in superposition. The bottom-left figure shows function λ with the zeros of γ in superposition. The bottom-right picture shows the function w/λ and the zeros of γ. It can be seen that the right-most zero minimizes the local error in each reported case.

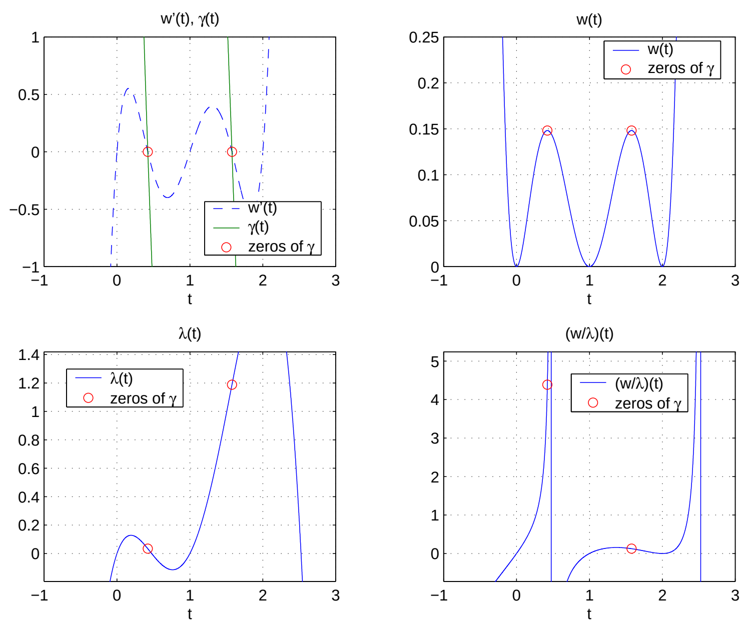

Figure 7.1: The functions γ, w, w', λ and w/λ for the case $k = 2, \sigma = (2, 2, 2)$.

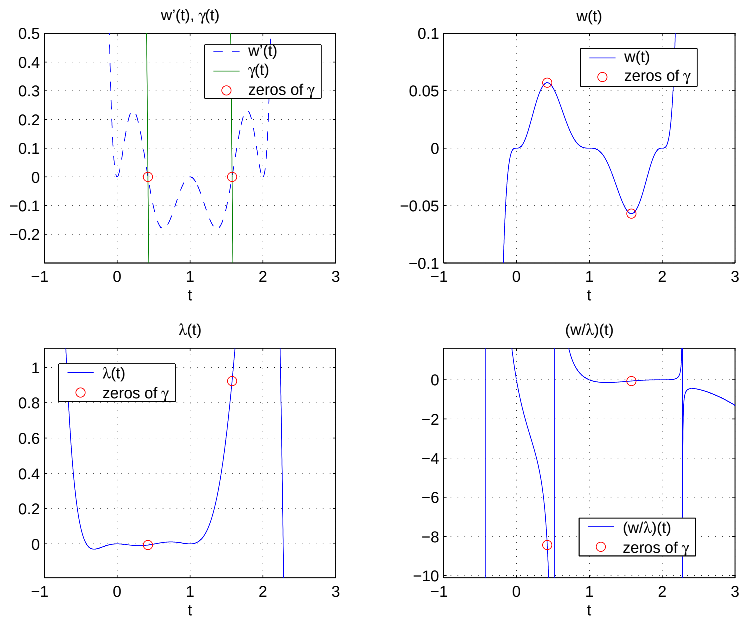

Figure 7.2: The functions γ, w, w', λ and w/λ for the case $k = 2, \sigma = (3, 3, 3)$.

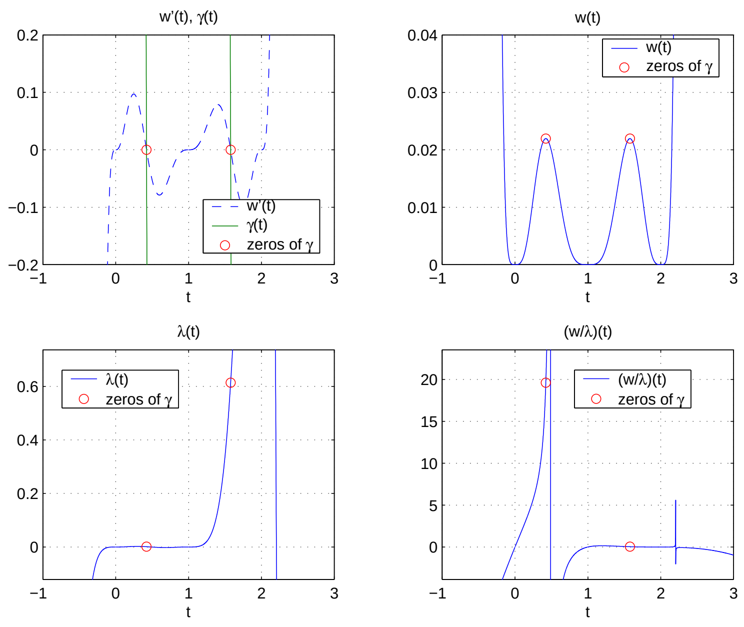

Figure 7.3: The functions γ, w, w', λ and w/λ for the case $k = 2, \sigma = (4, 4, 4)$.

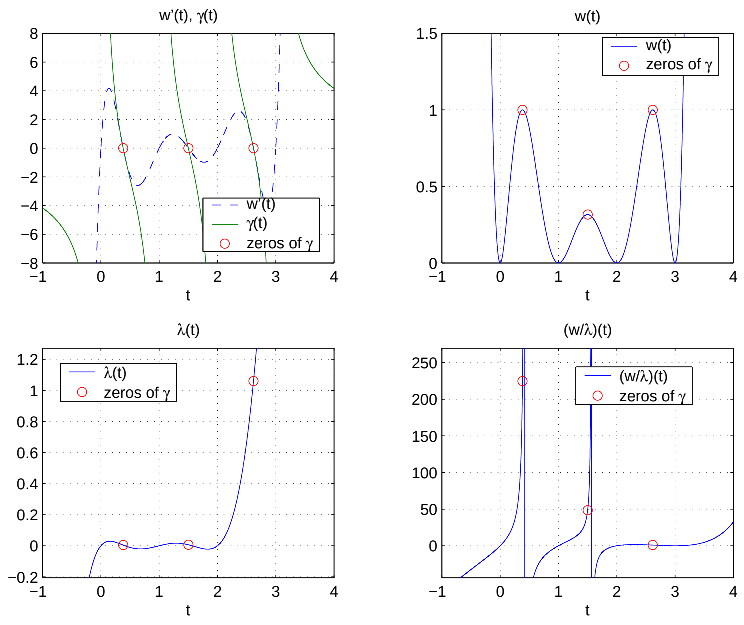

Figure 7.4: The functions γ, w, w', λ and w/λ for the case $k = 3, \sigma = (2, 2, 2, 2)$.

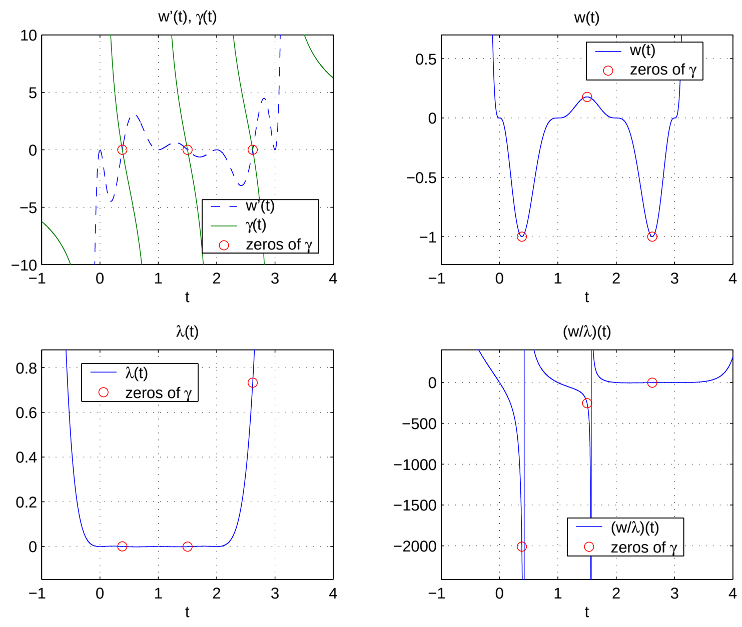

Figure 7.5: The functions γ, w, w', λ and w/λ for the case $k = 3, \sigma = (3, 3, 3, 3)$.

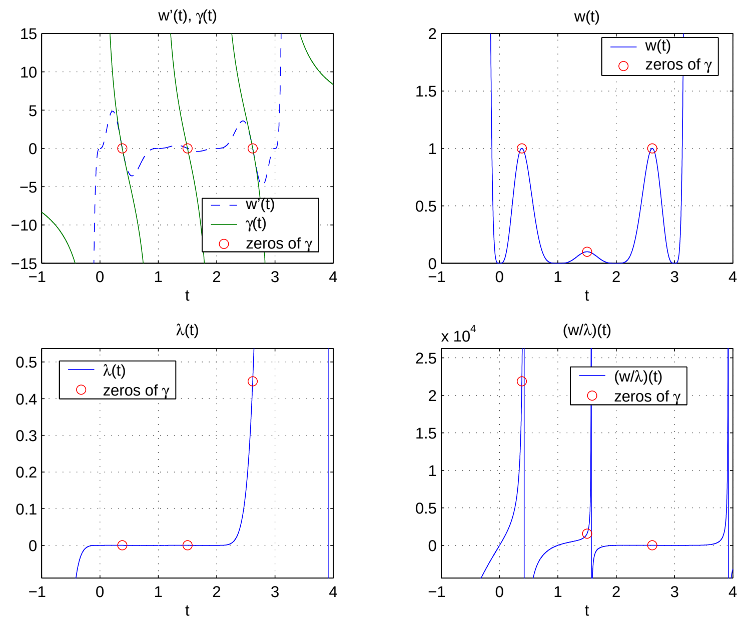

Figure 7.6: The functions γ, w, w', λ and w/λ for the case $k = 3, \sigma = (4, 4, 4, 4)$.

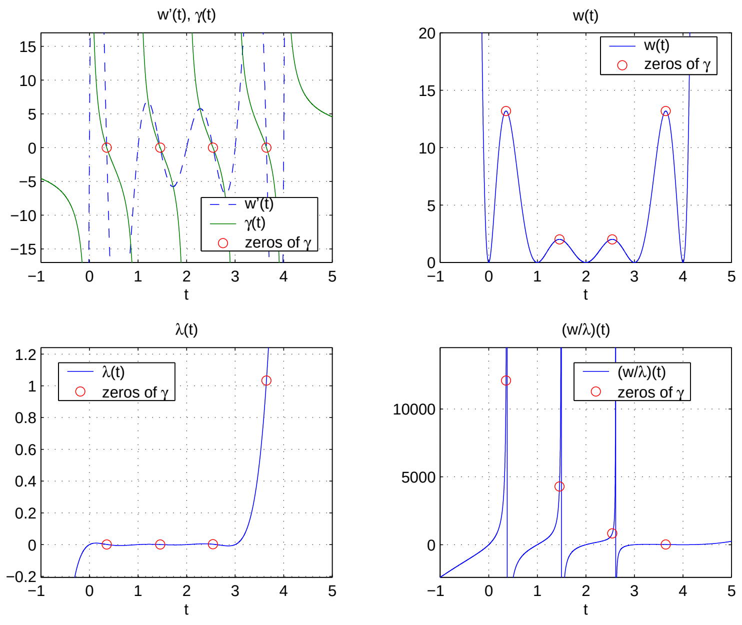

Figure 7.7: The functions γ, w, w', λ and w/λ for the case $k = 4, \sigma = (2,2,2,2,2)$.

7.7 Validity of the Asymptotic Assumption

Our analysis is based on the assumption that the step size h is sufficiently small. But *how small is sufficiently small?* According to our experiments, the actual step sizes are generally small enough so that the *asymptotically* optimal evaluation times produced by the above theory are good approximations of the *real* optima. There are two reasons for these small actual step sizes:

1. The need to bound the local error, which deteriorates the stability of validated methods and makes them less suited for stiff problems;

2. Current techniques implementing the bounding box process (see Section 3.1), which often impose a severe restriction on the step size, especially for stiff problems.

Example

Figure 7.8 illustrates our theoretical results experimentally on a specific ODE. It plots the local error of several global Hermite filters (GHF) as a function of the evaluation time for the Lorenz system (e.g., [HNW87]). It is assumed that $t_{i+1} - t_i$ is constant ($0 \leq i \leq 2k-2$). In addition, we assume that, in each mean-value filter composing the GHF, the distance between the evaluation time and

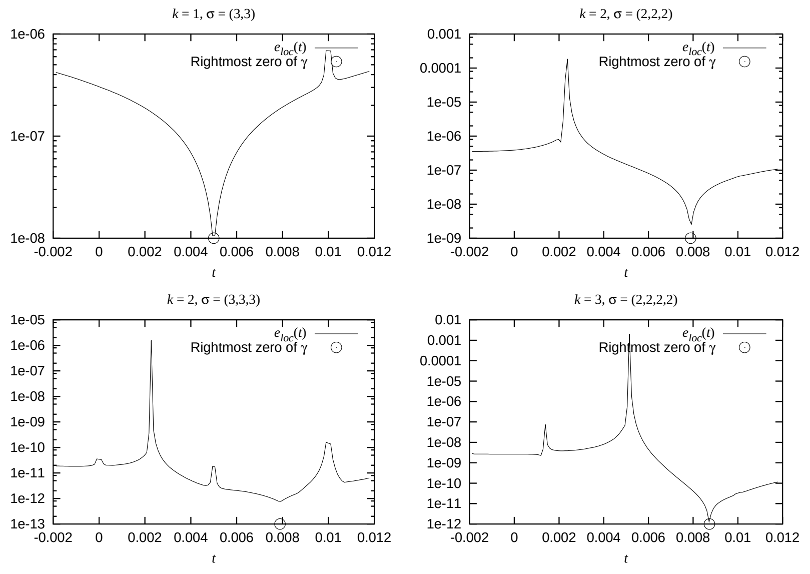

Figure 7.8: Local Error of Global Hermite Filters as a Function of the Evaluation Time for the Lorentz System.

the rightmost interpolation point is constant. In the graphs, $[t_0, t_k] = [0, 0.01]$ and $h = t_k - t_0 = 0.01$. The figure also shows the rightmost zero of the function γ as obtained from Table 7.1. As we can see, the rightmost zero of γ is a very good approximation of the optimal evaluation time of the filter for all the cases displayed.

Chapter 8

The Algorithm

We are now in position to present our algorithm for enclosing solutions of initial value problems for parametric ordinary differential equations. The algorithm is presented in Figure 8.1 and Figure 8.2 gives the specification of the functions not covered so far. The first two lines initialize the integration process and compute the initial bounding boxes, pruned domains, and the boxes and matrices needed for the wrapping effect. The main step of the integration are the lines 4-6. Line 4 computes the new bounding boxes, line 5 uses them to compute the new predicted boxes, and line 6 applies the pruning step to compute the new pruned boxes.

```
function SOLVE($\mathbb{O}, D_0, \mathbf{t}_{0..mk-1}$)
   begin
1     $\mathbf{B}_0 :=$ BOUNDINGBOX($\mathbb{O}, t_0, D_0, \mathbf{t}_0$);
2     $\langle \mathbf{D}_0^*, \mathbf{Y}_0, M_0 \rangle :=$ INITIALIZEMULTISTEP($\mathbb{O}, \mathbf{t}_0, D_0, \mathbf{B}_{1..k-1}$);
3     for $i := 1$ to $m-1$ do
4         $\mathbf{B}_i :=$ BOUNDINGBOX($\mathbb{O}, t_{ik-1}, D_{ik-1}^*, \mathbf{t}_i$);
5         $\mathbf{D}_i^- :=$ PREDICTOR($\mathbb{O}, t_{ik-1}, D_{ik-1}^*, \mathbf{t}_i, \mathbf{B}_i$);
6         $\langle \mathbf{D}_i^*, \mathbf{Y}_i, M_i \rangle :=$ PRUNE($\mathbb{O}, \mathbf{t}_{i-1}, \mathbf{D}_{i-1}^*, \mathbf{B}_{(i-1)k+1..ik-1}, \mathbf{Y}_{i-1}, M_{i-1}$,
                                        $\mathbf{t}_i, \mathbf{D}_i^-, \mathbf{B}_i$);
7     endfor
8     return $\mathbf{D}_{1..mk-1}^*$;
   end
```

Figure 8.1: The Constraint Satisfaction Algorithm for Initial Value Problems for Parametric ODEs.

Specification 4 (SOLVE) *Let s be the solution of ODE $\mathbb{O}$ and* $\mathbf{D}_{1..mk-1}$ = SOLVE($\mathbb{O}, D_0, \mathbf{t}_{0...mk-1}$). *Then, for $1 \leq i \leq mk-1$, $s(t_0, D_0, t_i) \subseteq D_i$.*

Specification 5 (BOUNDINGBOX) *Let* $\mathbf{B}_{1..k}$ =BOUNDINGBOX($\mathbb{O}, t_0, D_0, \mathbf{t}_{1..k}$). *Then, for $1 \leq i \leq k$, B_i is a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0).*

Specification 6 (INITIALIZEMULTISTEP) *Let ms be the multistep solution of ODE $\mathbb{O}$ and B_i be a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0) for $1 \leq i \leq k-1$. Let*

$$\langle \mathbf{D}_0, \mathbf{Y}_0, M \rangle = \text{INITIALIZEMULTISTEP}(\mathbb{O}, \mathbf{t}_0, D_0, \mathbf{B}_{1..k-1}) \tag{8.1}$$

and $\mathcal{A} = \{M\mathbf{y}_0 + m(\mathbf{D}_0) \mid \mathbf{y}_0 \in \mathbf{Y}_0\} \cap \mathbf{D}_0$. Then, for $0 \leq i \leq k-1$, $ms(t_0, D_0, t_i) \subseteq ms(\mathbf{t}_0, \mathcal{A}, t_i)$.

Specification 7 (PREDICTOR) *Let s be the solution of ODE $\mathbb{O}$ and B_i a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ wrt (t_0, D_0), for $1 \leq i \leq k$. Let*

$$\mathbf{D}_{1..k} = \text{PREDICTOR}(\mathbb{O}, t_0, D_0, \mathbf{t}_{1..k}, \mathbf{B}_{1..k}). \tag{8.2}$$

Then, for $1 \leq i \leq k$, $s(t_0, D_0, t_i) \subseteq D_i$.

Figure 8.2: The Specification of the Main Functions.

Chapter 9

Theoretical Analysis

This chapter presents theoretical results on the efficiency of our method and compares it to the best interval methods we are aware of.

9.1 Overview of the Methods

We analyze the cost of our SOLVE algorithm based on the global Hermite filter method GHF and compare it to Nedialkov's IHO method (see Section 3.4), the best interval method we know of. Indeed, the IHO method outperforms interval Taylor series methods such as Lohner's method (see Section 3.3). Here are the various methods used in the theoretical and experimental comparisons.

9.1.1 The GHF Method

In the GHF method, each iteration in the loop of function SOLVE is called a *step* of the integration. The (constant) step size in GHF is given by $h = t_k - t_0$. Assuming that $\sigma_m = \max(\sigma)$ and $\sigma_s = \sigma_0 + \ldots + \sigma_k$, the remaining components of GHF are specified as follows:

1. The BOUNDINGBOX function in GHF uses a Taylor series method (see Subsection 3.1.5) of order $p + q + 1$ to compute $\mathbf{B}_i$. Moreover, we assume that $B_{ik} = \ldots = B_{(i+1)k-1}$, i.e., the function computes a single bounding box over $[t_{ik-1}, t_{(i+1)k-1}]$ $(i \geq 1)$;

2. The PREDICTOR function uses Moore's Taylor method [Moo66] of order $q+1$ to compute the boxes $\mathbf{D}_i^-$. Note that we compute the Taylor coefficients of f only once at (t_{ik-1}, D_{ik-1}^*);

3. The evaluation point in Hermite filters (i.e., in function EMVFL) is the rightmost zero of function γ (see Chapter 7 and Table 7.1). GHF(σ) is thus a method of order $\sigma_s + 1$;

4. Function EXPLICITGLOBALFILTER needs $\sigma_m - 1$ Jacobians (i.e., $\mathcal{J}(D_j)_1, \ldots, \mathcal{J}(D_j)_{\sigma_m - 1}$) at each interpolation point t_j for $(i-1)k \leq j \leq (i+1)k - 1$ to compute the k explicit mean-value Hermite filters in EMVFL. GHF only computes Jacobians at predicted boxes and not at pruned boxes. More precisely, it only computes $k(\sigma_m - 1)$ Jacobians at $(\mathbf{t}_i, \mathbf{D}_i^-)$ and reuses the $k(\sigma_m - 1)$ Jacobians at $(\mathbf{t}_{i-1}, \mathbf{D}_{i-1}^-)$ which were computed during the previous step $i - 1$;

5. The function COORDTRANSFO uses Lohner's QR-factorization technique (see Subsection 3.3.4);

6. The function INITIALIZEMULTISTEP uses a one-step mean-value Taylor method (see Subsection 3.3.2).

9.1.2 The IHO Method

The IHO method is implemented exactly as described in [NJ99]. Its step size is h as in the GHF method. Besides the pruning, there are some interesting differences between GHF and IHO. First, the predictor function in IHO uses a mean-value Taylor method of order $q + 1$. Second, the Jacobians in IHO are recomputed at pruned boxes. IHO uses a Taylor series method of order $p+q+1$ to compute a bounding box as in GHF.

9.1.3 The IHO* Method

To obtain experimental results as informative as possible, we introduce IHO*, a variant of IHO which is closer to GHF. In particular, the predictor in IHO* uses Moore's Taylor method of order $q + 1$ instead of the mean-value Taylor method of the same order. Also, IHO* does not recompute the Jacobians at pruned boxes; it reuses the Jacobians at predicted boxes instead as in GHF. These two changes make IHO* and GHF closer; they only differ in the pruning step. Interestingly, IHO* is extremely close in precision to IHO on almost all benchmarks for a given step size. There are a few benchmarks where the loss of precision is significant or where a smaller step size must be used. Of course, IHO* is faster than IHO for a given step size.

9.2 Comparison Hypotheses

We make the following assumptions and conventions for simplicity. Consider the ODE $u' = f(u)$. We assume that (the natural encoding of) function f contains only arithmetic operations. We denote by N_1 the number of $*, /$ operations in f, by N_2 the number of $\pm$ operations, and by N the sum $N_1 + N_2$. We also assume that the cost of evaluating $\mathcal{J}(D_i)_j$ is n times the cost of evaluating $(D_i)_j$. We report only the *main* operations of the methods, i.e., (1) products of a real and an interval matrix which arise in the pruning step and (2) the generation of

Jacobians [1]. These are the main operations for problems of sufficiently high dimension where f contains sufficiently many operations. Note that products of a real and an interval matrix can be optimized to substantially reduce the number of sign tests and rounding mode switches, which are costly tasks (see [Knu94]). As a consequence, the cost *per* interval arithmetic operation in a real-interval matrix product is less than the cost of an operation on two intervals in a Jacobian computation. We thus report separately the number of interval arithmetic operations involved in products of a real and an interval matrix in the pruning step (Cost-1) and the generation of Jacobians (Cost-2). Note that Cost-1 is a fixed cost in the sense that it is independent from the ODE. Cost-2 is a variable cost which increases as the expression of f contains more operations.

9.3 Methods of the Same Order

We first compare the costs of GHF(σ) and IHO$^{(*)}(p,q)$ for $p+q=\sigma_s$ and $q \in \{p, p+1\}$. The methods are thus of order $\sigma_s + 1$. Table 9.1 reports the main cost of a step in IHO, IHO*, and GHF (see Appendix B). It also shows the complexity of two particular cases of GHF: GHF-1 is an implementation with only two interpolation points ($k=1$) and $|\sigma_1 - \sigma_0| \leq 1$, while GHF-2 is an implementation with two conditions on every interpolation points ($\sigma_0 = \ldots = \sigma_k = 2$).

The first main result is that GHF-1 is always cheaper than IHO$^{(*)}$. *Hence a GHF method with only two interpolation points is guaranteed to run faster than IHO$^{(*)}$.* The next chapter shows that an improvement in accuracy is also obtained in this case. Observe that Cost-2 in IHO* is approximately half as much as in IHO because the Jacobians are not computed at pruned boxes in IHO*. Note also that Cost-2 is smaller in GHF-1 than in IHO* because IHO* evaluates one more Jacobian, i.e., $\mathcal{J}(D_i)_q$.

GHF-2 is more expensive than GHF-1 and IHO$^{(*)}$ when f contains few operations because the Jacobians are cheap to compute in this case and the fixed cost Cost-1 becomes large wrt Cost-2. However, when f contains many $*, /$ operations (which is the case in many practical applications), GHF-2 becomes substantially faster because Cost-1 in GHF-2 is independent of f and Cost-2 is substantially smaller in GHF-2 than in GHF-1 and IHO$^{(*)}$. *This result shows the versatility of the approach that can be taylored to the application at hand.*

9.3.1 One-Step Methods of Different Orders but of Similar Cost

We now show that GHF methods can be tailored to be asymptotically more precise than IHO methods for a similar cost. Consider the costs of the IHO$^{(*)}(p,q)$ and GHF-1 methods when we assume that $p+q=\sigma_s-2$ and $q \in \{p,p+1\}$. Under these conditions, IHO$^{(*)}$ is a method of order $\sigma_s - 1$, while GHF-1 is a

[1]Matrix inversions and the QR-factorization in CoordTransfo are not counted here.

	Cost-1	Cost-2
IHO	–	$2\lceil\frac{\sigma_s}{2}\rceil^2 nN_1 + O(\sigma_s nN_2)$
IHO*	–	$\lceil\frac{\sigma_s}{2}\rceil^2 nN_1 + O(\sigma_s nN_2)$
GHF	$7k^3n^3$	$((\sigma_m - 1)^2 + 1)knN_1 + \sigma_m knN_2$
GHF-1	–	$(\lfloor\frac{\sigma_s-1}{2}\rfloor^2 + 1)nN_1 + O(\sigma_s nN_2)$
GHF-2	$(\frac{7}{8}\sigma_s - \frac{21}{4})\sigma_s^2 n^3$	$(\sigma_s - 2)nN$

Table 9.1: Cost Analysis : Methods of the Same Order.

	Cost-2
IHO	$2\lfloor\frac{\sigma_s-1}{2}\rfloor^2 nN_1 + O(\sigma_s nN_2)$
IHO*	$\lfloor\frac{\sigma_s-1}{2}\rfloor^2 nN_1 + O(\sigma_s nN_2)$
GHF-1	$(\lfloor\frac{\sigma_s-1}{2}\rfloor^2 + 1)nN_1 + O(\sigma_s nN_2)$

Table 9.2: Cost Analysis : Methods of Different Orders but of Similar Cost.

method of order $\sigma_s + 1$. Table 9.2 reports the main cost of a step in IHO, IHO*, and GHF-1. Cost-2 is similar in GHF-1 and IHO* (and about twice as much in IHO). The GHF-1 method is thus asymptotically more precise (by two orders of magnitude) than IHO* for a similar cost.

Chapter 10

Experimental Analysis

10.1 Overview

We now report experimental results of a C++ implementation [1] of our SOLVE algorithm based on the global Hermite filter method GHF(σ). We performed our tests on a Sun Ultra 10 workstation with a 333 MHz UltraSparc CPU. The underlying interval arithmetic and automatic differentiation packages are PROFIL/BIAS [Knu94] and FADBAD/TADIFF [BS96, BS97] respectively.

10.1.1 The Benchmarks

Many of the benchmarks are standard for ODE solvers. They come from various domains, including chemistry, biology, mechanics, physics and electricity. The equation, initial conditions, and interval of integration for each initial value problem are given in Appendix A. Note that the comparisons only uses point initial conditions; they could easily be generalized to interval conditions. The "full Brusselator" (BRUS), the "Oregonator" (OREG), and HIRES all model famous chemical reactions. Both OREG and HIRES are stiff problems. The Lorenz system (LOR) examplifies the so-called "strange attractors". the Two-Body problem (2BP) comes from mechanics and the van der Pol (VDP) equation describes an electrical circuit. All these problems are described in detail in [HNW87, HW91]. We also consider a problem from molecular biology (BIO) and the Stiff DETEST problem D1 [Enr75]. Finally, we consider four dynamical systems (LIEN, P1, P2, P3), where the function f contains more operations. LIEN, P2 and P3 are taken from [Per00].

10.1.2 Overview of the Experiments

The experimental results obey the same assumptions as the theoretical analysis. They include three types of comparisons :

[1] The code is available at `http://www.info.ucl.ac.be`.

1. One-step methods of the same order;
2. One-step methods of different orders, but of similar cost;
3. Multistep versus one-step methods of the same order.

The tables report, for a given step size, the global error, the error ratio (an error ratio higher than 1 means that GHF is more precise), the execution time of both methods (in seconds), and the time ratio (a time ratio higher than 1 means that GHF is faster). They also report the execution time of IHO* between parentheses. As mentioned, we observed small precision loss in IHO* over IHO and only for the larger step sizes. Since this was not very significant, we assume that the error values in IHO* are nearly the same as in IHO. A "-" symbol in the tables means that the method failed to integrate the ODE for the corresponding step size. Finally, note that the global error at point t_i is given by the infinity norm of the width of the enclosure D_i at t_i, i.e., the quantity $\|\omega(D_i)\|_\infty$ at the end of the interval of integration.

10.2 One-Step Methods

10.2.1 Same Order

Table 10.1 reports the experimental results for the $\mathrm{IHO}^{(*)}(p,p)$ and $\mathrm{GHF}(p,p)$ methods of order $2p+1$ on several benchmarks, orders, and step sizes. In general, for a given step size, GHF and IHO* have a similar accuracy and execution time. GHF is usually slightly faster as predicted by the theoretical results. The difference should be larger for higher dimensional problems where f contains many operations. IHO is slower than GHF and IHO*. For a given problem and given order, the error ratio is generally constant wrt the step size, confirming that GHF and $\mathrm{IHO}^{(*)}$ are methods of the same order.

10.2.2 Different Orders

The theoretical results indicated that, given a step size, the GHF method can always be tailored to be asymptotically more precise than IHO* for a similar computation cost. We now validate this claim experimentally. Table 10.2 compares $\mathrm{IHO}(p,p)$ (order $2p+1$) and $\mathrm{GHF}(p+1,p+1)$ (order $2p+3$). On the benchmarks, GHF is always faster than IHO and it produces significant improvements in accuracy. As expected, the gain in precision increases when the step size decreases confirming that GHF is a method of higher order than IHO. GHF is slightly slower than IHO* but, of course, it produces significant improvement in accuracy. GHF and IHO* should have a similar execution time for higher dimensional problems where f contains many operations, as predicted by the theoretical analysis.

IVP	IHO	GHF	h	Error			Time		
	p, q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
BRUS	3,3	(3,3)	1E-1	2.3E-3	1.2E-3	1.9			
			7.5E-2	4.5E-5	2.4E-5	1.9			
			5E-2	9.7E-7	4.9E-7	2.0			
			2.5E-2	5.2E-9	2.7E-9	1.9			
			1.25E-2	3.2E-11	1.7E-11	1.9			
			1E-2	6.5E-12	3.5E-12	1.9	5.1 (4.0)	3.9	1.3
	4,4	(4,4)	1E-1	1.7E-4	9.9E-5	1.7			
			7.5E-2	2.0E-6	1.1E-6	1.8			
			5E-2	1.0E-8	5.0E-9	2.0			
			2.5E-2	7.4E-12	3.2E-12	2.3	2.8 (2.1)	2.0	1.4
	5,5	(5,5)	1E-1	2.4E-5	1.6E-5	1.5			
			7.5E-2	1.2E-7	7.6E-8	1.6			
			5E-2	1.6E-10	9.4E-11	1.7	1.9 (1.4)	1.3	1.5
	7,7	(7,7)	1E-1	7.6E-7	5.2E-7	1.5			
			7.5E-2	6.6E-10	4.7E-10	1.4	1.9 (1.4)	1.3	1.5
	8,8	(8,8)	1E-1	1.5E-7	1.1E-7	1.4			
			7.5E-2	5.4E-11	4.0E-11	1.4	2.2 (1.6)	1.5	1.5
LOR	3,3	(3,3)	1.25E-2	4.8E-1	3.2E-1	1.5			
			1E-2	6.7E-2	4.5E-2	1.5			
			7.5E-3	7.7E-3	4.9E-3	1.6			
			5E-3	4.3E-4	2.6E-4	1.7			
			2.5E-3	3.1E-6	2.0E-6	1.6	11 (8)	8	1.4
	4,4	(4,4)	2E-2	1.5E-1	1.0E-1	1.5			
			1.75E-2	2.7E-2	1.8E-2	1.5			
			1.5E-2	5.0E-3	3.0E-3	1.7			
			1.25E-2	8.0E-4	4.6E-4	1.7			
			1E-2	9.0E-5	5.0E-5	1.8			
			7.5E-3	6.0E-6	3.1E-6	1.9	4.7 (3.6)	3.6	1.3
	7,7	(7,7)	3E-2	3.0E-3	2.4E-3	1.2			
			2.75E-2	4.5E-4	3.6E-4	1.2			
			2.5E-2	6.6E-5	5.3E-5	1.2			
			2.25E-2	7.7E-6	6.2E-6	1.2	3.0 (2.3)	2.2	1.4
2BP	3,3	(3,3)	1E-1	4.5E-3	7.6E-4	6.0			
			7.5E-2	1.1E-4	3.7E-5	3.0			
			5E-2	3.3E-6	1.2E-6	2.7			
			2.5E-2	1.5E-8	4.5E-9	3.3	3.6 (2.9)	2.6	1.4
	4,4	(4,4)	1.25E-1	2.9E-4	7.4E-5	3.9			
			1E-1	1.2E-5	3.0E-6	4.0			
			7.5E-2	3.4E-7	8.5E-8	4.0			
			5E-2	3.4E-9	9.2E-10	3.7	2.5 (1.9)	1.7	1.5
	7,7	(7,7)	1.5E-1	1.1E-6	5.6E-7	2.0			
			1.25E-1	2.3E-9	9.7E-10	2.4	2.0 (1.5)	1.3	1.5
VDP	3,3	(3,3)	4E-2	1.5E-2	5.8E-3	2.6			
			3E-2	5.9E-5	3.8E-5	1.6			
			2E-2	1.7E-6	9.6E-7	1.8			
			1E-2	1.0E-8	5.3E-9	1.9			
			5E-3	7.4E-11	3.8E-11	1.9			
			2.5E-3	4.7E-13	2.6E-13	1.8	14 (11.2)	11.6	1.2
	4,4	(4,4)	4E-2	4.7E-5	4.0E-5	1.2			
			3E-2	8.4E-7	5.1E-7	1.6			
			2E-2	9.0E-9	4.5E-9	2.0			
			1E-2	1.1E-11	4.7E-12	2.3	4.5 (3.7)	3.8	1.2
	5,5	(5,5)	4E-2	2.6E-6	2.1E-6	1.2			
			3E-2	2.3E-8	1.6E-8	1.4			
			2E-2	6.7E-11	3.9E-11	1.7	2.9 (2.3)	2.4	1.3
BIO	3,3	(3,3)	7.5E-3	4.6E-6	2.0E-6	2.3			
			5E-3	8.2E-9	3.4E-9	2.4			
			2.5E-3	2.2E-11	9.2E-12	2.4	7.0 (5.4)	5.1	1.4
	4,4	(4,4)	7.5E-3	1.3E-6	7.6E-7	1.7			
			5E-3	2.9E-10	1.3E-10	2.2			
			2.5E-3	9.7E-14	3.3E-14	2.9	10 (7.5)	7.0	1.4
OREG	3,3	(3,3)	1.5E-2	1.5E-4	2.2E-4	0.7			
			1E-2	8.0E-6	1.1E-5	0.7			
			7.5E-3	1.0E-6	1.4E-6	0.7			
			5E-3	6.0E-8	7.9E-8	0.8	9.6 (7.7)	7.5	1.3
	4,4	(4,4)	2.5E-2	2.4E-4	3.4E-4	0.7			
			2E-2	1.2E-5	1.6E-5	0.7			
			1.5E-2	6.1E-7	7.6E-7	0.8			
			1E-2	1.5E-8	1.9E-8	0.8			
			7.5E-3	1.1E-9	1.4E-9	0.8	8.2 (6.5)	6.4	1.3
D1	8,8	(8,8)	1.1E-1	1.1E-6	1.3E-6	0.8			
			1E-1	1.3E-7	1.4E-7	0.9			
			9E-2	1.5E-8	1.7E-8	0.9			
			8E-2	1.5E-9	1.7E-9	0.9			
			7E-2	1.3E-10	1.4E-10	0.9			
			6E-2	7.3E-12	8.3E-12	0.9			
			5E-2	2.8E-13	3.1E-13	0.9	2.4 (1.8)	1.9	1.3
HIRES	4,4	(4,4)	2.5E-1	3.2E-7	6.1E-7	0.5			
			2E-1	2.4E-8	4.3E-8	0.6			
			1.5E-1	1.1E-9	2.6E-9	0.4			
			1E-1	2.8E-11	5.0E-11	0.6			
			5E-2	4.8E-14	6.9E-14	0.7	23 (17)	16	1.4
	8,8	(8,8)	4E-1	2.9E-6	1.2E-5	0.2			
			3.5E-1	4.9E-8	3.9E-8	1.3			
			3E-1	8.0E-10	6.2E-10	1.3			
			2.5E-1	7.7E-12	6.0E-12	1.3			
			2E-1	3.4E-14	2.8E-14	1.2	10.9 (7.4)	7.2	1.5

Table 10.1: One-Step Methods of the Same Order.

IVP	IHO	GHF	h	Error			Time		
	p, q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
BRUS	3,3	(4,4)	1E-1	2.3E-3	1.0E-3	2.3			
			7.5E-2	4.5E-5	1.3E-5	3.5			
			5E-2	9.7E-7	1.2E-7	8.1			
			2.5E-2	5.2E-9	9.5E-11	55			
			1.25E-2	3.2E-11	2.0E-13	160	4.0 (3.2)	3.6	1.1
	4,4	(5,5)	1E-1	1.7E-4	1.0E-4	1.7			
			7.5E-2	2.0E-6	9.9E-7	2.0			
			5E-2	1.0E-8	3.2E-9	3.1			
			2.5E-2	7.4E-12	6.4E-13	12	2.8 (2.1)	2.4	1.2
LOR	3,3	(4,4)	1.25E-2	4.8E-1	1.3E-2	1.5			
			1E-2	6.7E-2	1.2E-3	56			
			7.5E-3	7.7E-3	5.7E-5	135			
			5E-3	4.3E-4	9.7E-7	443	5.4 (4.0)	4.9	1.1
	4,4	(5,5)	2E-2	1.5E-1	6.2E-2	2.4			
			1.75E-2	2.7E-2	9.0E-3	3.0			
			1.5E-2	5.0E-3	1.2E-3	4.2			
			1.25E-2	8.0E-4	1.2E-4	6.7			
			1E-2	9.0E-5	7.2E-6	13			
			7.5E-3	6.0E-6	2.6E-7	23	4.7 (3.6)	4.1	1.1
2BP	3,3	(4,4)	1E-1	4.5E-3	2.5E-5	180			
			7.5E-2	1.1E-4	7.6E-7	145			
			5E-2	3.3E-6	8.9E-9	371			
			2.5E-2	1.5E-8	4.1E-11	366	3.6 (2.9)	3.0	1.2
	4,4	(5,5)	1.25E-1	2.9E-4	1.1E-5	26			
			1E-1	1.2E-5	3.6E-7	33			
			7.5E-2	3.4E-7	5.6E-9	61			
			5E-2	3.4E-9	5.5E-11	62	2.5 (1.9)	2.0	1.3
VDP	3,3	(4,4)	4E-2	1.5E-2	2.5E-3	6.0			
			3E-2	5.9E-5	9.7E-6	6.1			
			2E-2	1.7E-6	8.8E-8	19			
			1E-2	1.0E-8	6.2E-11	161			
			5E-3	7.4E-11	9.0E-14	822	7.4 (5.6)	7.2	1.0
	4,4	(5,5)	4E-2	4.7E-5	3.6E-5	1.3			
			3E-2	8.4E-7	3.6E-7	2.3			
			2E-2	9.0E-9	1.6E-9	5.6			
			1E-2	1.1E-11	2.8E-13	39	4.5 (3.7)	4.2	1.1
BIO	3,3	(4,4)	7.5E-3	4.6E-6	1.7E-6	2.7			
			5E-3	8.2E-9	1.2E-9	6.8			
			2.5E-3	2.2E-11	4.8E-13	46	7.0 (5.4)	6.2	1.1
	4,4	(5,5)	7.5E-3	1.3E-6	7.7E-7	1.7			
			5E-3	2.9E-10	9.3E-11	3.1			
			2.5E-3	9.7E-14	1.0E-14	9.7	10 (7.5)	8.4	1.2
OREG	3,3	(4,4)	2E-2	2.6E-3	7.0E-5	37			
			1.5E-2	1.5E-4	1.1E-6	136			
			1E-2	8.0E-6	2.2E-8	364			
			7.5E-3	1.0E-6	1.5E-9	667			
			5E-3	6.0E-8	4.6E-11	1304	9.6 (7.7)	8.6	1.1
	4,4	(5,5)	2.5E-2	2.4E-4	1.4E-4	1.7			
			2E-2	1.2E-5	3.9E-6	3.1			
			1.5E-2	6.1E-7	1.6E-8	38			
			1E-2	1.5E-8	6.3E-11	238	6.2 (4.9)	5.3	1.2
D1	8,8	(9,9)	1.1E-1	1.1E-6	3.9E-8	28			
			1E-1	1.3E-7	3.6E-9	36			
			9E-2	1.5E-8	3.5E-10	43			
			8E-2	1.5E-9	2.9E-11	53			
			7E-2	1.3E-10	1.8E-12	72			
			6E-2	7.3E-12	7.8E-14	94	2.0 (1.5)	1.8	1.1
HIRES	4,4	(5,5)	3E-1	1.3E-5	1.9E-6	6.8			
			2.5E-1	3.2E-7	6.0E-8	5.3			
			2E-1	2.4E-8	2.4E-9	10			
			1.5E-1	1.1E-9	4.6E-11	24			
			1E-1	2.8E-11	3.2E-13	88	12 (8.5)	9.3	1.3
	8,8	(9,9)	4E-1	2.9E-6	2.5E-5	0.1			
			3.5E-1	4.9E-8	4.1E-8	1.2			
			3E-1	8.0E-10	6.5E-10	1.2			
			2.5E-1	7.7E-12	6.2E-12	1.2			
			2E-1	3.4E-14	2.9E-14	1.2	10.9 (7.4)	7.9	1.4

Table 10.2: One-Step Methods of Different Orders.

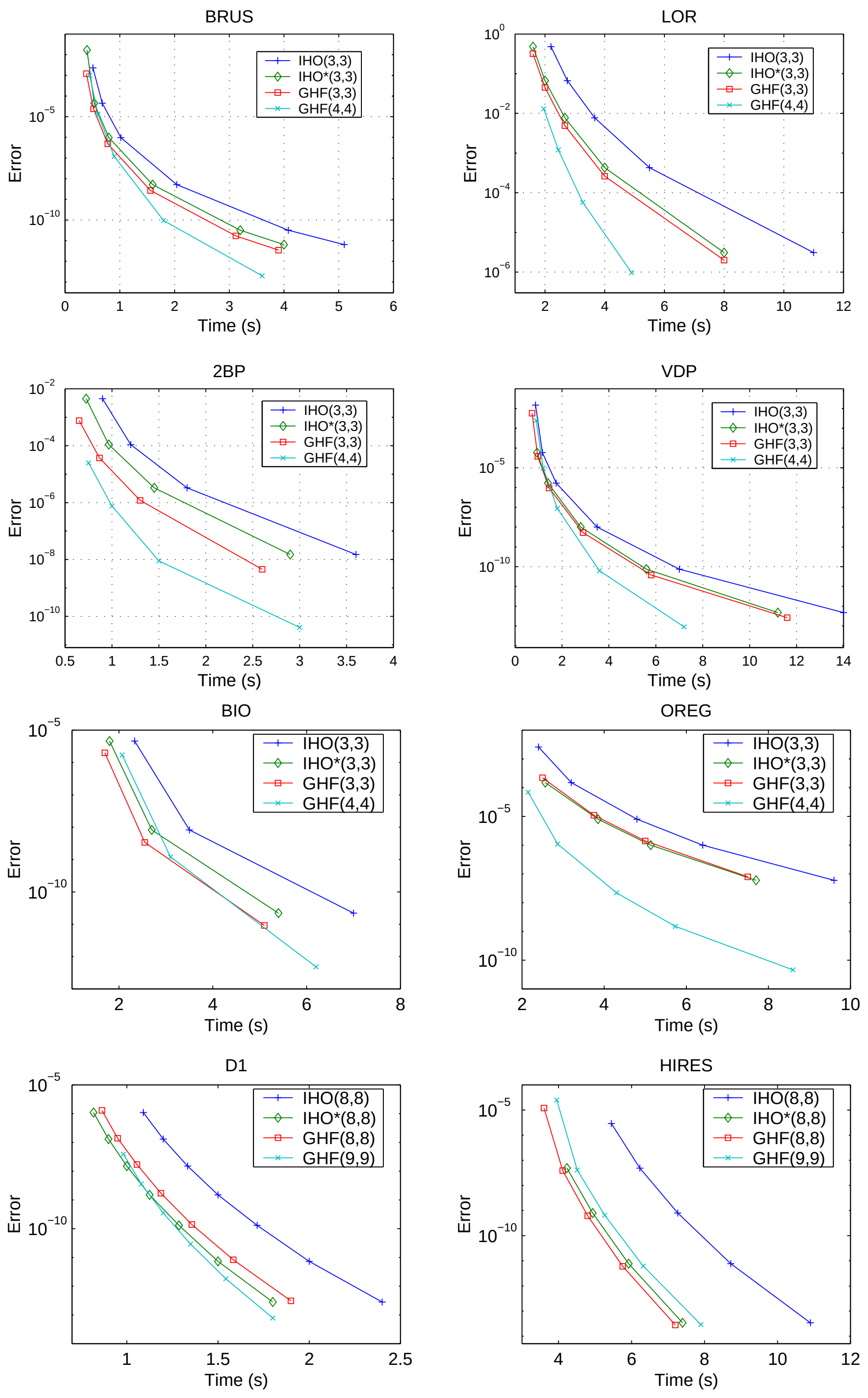

Figure 10.1: Comparison of the Methods IHO$^{(*)}(p,p)$, GHF(p,p) and GHF$(p+1,p+1)$ for the Problems BRUS, LOR, 2BP, VDP, BIO, OREG, D1 and HIRES.

10.2.3 Error wrt Time

It is interesting to compare the various methods by plotting the error as a function of the execution time. Figure 10.1 plots IHO$^{(*)}(p,p)$, GHF(p,p), and GHF$(p+1,p+1)$ using the results in Tables 10.1 and 10.2. We take $p=8$ for D1 and HIRES and $p=3$ for the other problems. The curve of IHO* is always slightly above the curve of GHF(p,p) (except for D1). GHF$(p+1,p+1)$ is almost always below the other curves and IHO is always above the other curves. These results confirm the theoretical results and indicate that GHF$(p+1,p+1)$ is superior to the other methods.

10.3 Multistep Versus One-Step Methods

We now compare multistep GHF methods versus IHO$^{(*)}$ and the one-step GHF method of the same order. We restrict attention to problems where the function f contains more operations. Tables 10.3, 10.4, 10.5, and 10.6 report the results respectively for the four tested examples and for several orders and step sizes [2]. For a given step size, multistep GHF methods usually produce much more precise results than one-step methods (especially for large step sizes); they also allow for larger step sizes. Multistep GHF methods are generally as fast as the one-step GHF method and IHO*; they are faster when f has many operations, as is the case in LIEN (which contains many multiplications). The tables also show that, for a given step size, the one-step GHF method is slightly more precise and faster than IHO*, and that IHO is slower.

Figures 10.2, 10.4, 10.3 and 10.5 plot the error as a function of the execution time. The main result is that multistep GHF methods perform better than one-step methods on these problems. In general, multistep methods produce several orders of magnitude improvements in precision for a fixed execution time. The one-step GHF method performs slightly better than IHO*. Note that, for the LIEN problem, GHF methods with many interpolation points are more efficient and allow for smaller execution times.

10.4 Discussion

Before concluding this chapter, it is important to make a number of remarks.

1. In GHF, the enhancement in precision obtained by recomputing the Jacobians at pruned boxes is insignificant in all problems we tested. Instead, this recomputation increases the computational cost. Our experimental results showed that this also holds for the IHO method in general.

2. As pointed out by Nedialkov [Ned99], the stability of interval methods depends not only on the stability of the underlying approximation formula

[2]Note that in the LIEN problem, we used a bounding box computation method of order 13 for $\sigma_s \geq 12$.

IVP	IHO	GHF	h	Error			Time		
	p,q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
LIEN	3,3	(3,3)	5E-1	8.8E-7	7.2E-7	1.2			
			4E-1	1.4E-8	1.1E-8	1.3			
			3E-1	8.4E-10	4.7E-9	0.2			
			2E-1	2.3E-11	5.4E-11	0.4			
			1E-1	1.3E-13	1.3E-13	1.0			
			5E-2	8.7E-16	8.8E-16	1.0	8.3 (6.7)	6.1	1.4
	3,3	(2,2,2)	5.5E-1	-	2.1E-6	-			
			5E-1	8.8E-7	2.5E-7	3.5			
			4E-1	1.4E-8	3.0E-9	4.7			
			3E-1	8.4E-10	1.9E-10	4.4			
			2E-1	2.3E-11	6.9E-12	3.3			
			1E-1	1.3E-13	3.7E-14	3.5			
			5E-2	8.7E-16	2.6E-16	3.3	8.3 (6.7)	6.3	1.3
	4,4	(4,4)	5E-1	2.5E-7	2.0E-7	1.3			
			4E-1	1.0E-9	7.6E-10	1.3			
			3E-1	1.9E-11	1.4E-11	1.4			
			2E-1	1.1E-13	8.3E-14	1.3			
			1E-1	8.3E-17	6.5E-17	3.5	6.1 (4.8)	4.4	1.4
	4,4	(2,2,2,2)	5.8E-1	-	6.7E-8	-			
			5.5E-1	-	8.5E-9	-			
			5E-1	2.5E-7	7.2E-9	35			
			4E-1	1.0E-9	5.0E-11	20			
			3E-1	1.9E-11	1.1E-12	17			
			2E-1	1.1E-13	9.9E-15	11			
			1E-1	8.3E-17	3.8E-17	2.2	6.1 (4.8)	4.6	1.3
	5,5	(5,5)	5E-1	1.2E-7	9.4E-8	1.3			
			4E-1	1.2E-10	9.1E-11	1.3			
			3E-1	7.4E-13	5.7E-13	1.3			
			2E-1	9.9E-16	7.2E-16	1.4	4.2 (3.3)	3.0	1.4
	5,5	(2,2,2,2,2)	5.8E-1	-	6.3E-9	-			
			5.5E-1	-	8.2E-10	-			
			5E-1	1.2E-7	9.3E-11	1290			
			4E-1	1.2E-10	2.0E-12	60			
			3E-1	7.4E-13	2.4E-14	31			
			2E-1	9.9E-16	1.0E-16	10	4.2 (3.3)	3.1	1.3
	6,6	(6,6)	5E-1	7.2E-8	6.0E-8	1.2			
			4.5E-1	3.5E-10	2.9E-10	1.2			
			4E-1	1.7E-11	1.4E-11	1.2			
			3.5E-1	9.1E-13	7.4E-13	1.2			
			3E-1	4.0E-14	3.3E-14	1.2	3.7 (2.9)	2.7	1.4
	6,6	(4,4,4)	5.5E-1	-	1.2E-7	-			
			5E-1	7.2E-8	2.0E-10	360			
			4.5E-1	3.5E-10	1.8E-11	19			
			4E-1	1.7E-11	1.3E-12	13			
			3.5E-1	9.1E-13	9.0E-14	10			
			3E-1	4.0E-14	3.9E-15	10	3.7 (2.9)	2.7	1.4
	6,6	(3,3,3,3)	5.8E-1	-	3.8E-8	-			
			5.5E-1	-	8.6E-10	-			
			5E-1	7.2E-8	3.9E-10	185			
			4.5E-1	3.5E-10	3.0E-11	12			
			4E-1	1.7E-11	6.0E-13	28			
			3.5E-1	9.1E-13	4.8E-14	19			
			3E-1	4.0E-14	2.3E-15	17	3.7 (2.9)	2.6	1.4
	6,6	(2,2,2,2,2,2)	6E-1	-	1.5E-8	-			
			5.5E-1	-	1.7E-10	-			
			5E-1	7.2E-8	1.4E-11	5143			
			4.5E-1	3.5E-10	1.6E-12	219			
			4E-1	1.7E-11	1.4E-13	121			
			3.5E-1	9.1E-13	1.3E-14	70			
			3E-1	4.0E-14	9.0E-16	44	3.7 (2.9)	2.8	1.3
	8,8	(8,8)	5E-1	2.5E-8	2.1E-8	1.2			
			4.5E-1	2.5E-11	2.1E-11	1.2			
			4E-1	5.1E-13	4.3E-13	1.2			
			3.5E-1	1.1E-14	9.7E-15	1.2			
			3E-1	2.0E-16	1.7E-16	1.2	5.1 (3.7)	3.4	1.5
	8,8	(4,4,4,4)	5.8E-1	-	1.2E-8	-			
			5.5E-1	-	8.9E-11	-			
			5E-1	2.5E-8	1.7E-11	1471			
			4.5E-1	2.5E-11	6.7E-13	37			
			4E-1	5.1E-13	6.8E-15	75			
			3.5E-1	1.1E-14	4.1E-16	27	4.4 (3.2)	2.8	1.6
	9,9	(9,9)	5E-1	1.5E-8	1.3E-8	1.2			
			4.5E-1	7.2E-12	6.2E-12	1.2			
			4E-1	9.7E-14	8.3E-14	1.2			
			3.5E-1	1.4E-15	1.2E-15	1.2	5.1 (3.7)	3.4	1.5
	9,9	(6,6,6)	5.5E-1	-	1.9E-8	-			
			5E-1	1.5E-8	5.3E-12	2830			
			4.5E-1	7.2E-12	1.6E-13	45			
			4E-1	9.7E-14	4.0E-15	24			
			3.5E-1	1.4E-15	1.3E-16	11	5.1 (3.7)	3.2	1.6
	9,9	(3,3,3,3,3,3)	6E-1	-	3.5E-7	-			
			5.5E-1	-	2.5E-11	-			
			5E-1	1.5E-8	7.5E-13	20000			
			4.5E-1	7.2E-12	2.2E-13	33			
			4E-1	9.7E-14	2.0E-14	4.8			
			3.5E-1	1.4E-15	2.5E-14	0.06	5.1 (3.7)	3.1	1.6

Table 10.3: Multistep Versus One-Step Methods : the LIEN Problem.

IVP	IHO	GHF	h	Error			Time		
	p, q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
P1	3,3	(3,3)	5E-2	8.1E-5	5.2E-5	1.6			
			4E-2	4.0E-6	2.3E-6	1.7			
			3E-2	2.0E-7	1.1E-7	1.8			
			2E-2	6.1E-9	2.9E-9	2.1			
			1E-2	3.3E-11	1.4E-11	2.4			
			5E-3	2.4E-13	9.7E-14	2.5	27 (22)	21	1.3
	3,3	(2,2,2)	6.5E-2	-	1.7E-4	-			
			6E-2	-	3.2E-5	-			
			5E-2	8.1E-5	2.9E-6	28			
			4E-2	4.0E-6	2.8E-7	14			
			3E-2	2.0E-7	2.1E-8	9.5			
			2E-2	6.1E-9	7.9E-10	7.8			
			1E-2	3.3E-11	4.0E-12	8.4			
			5E-3	2.4E-13	3.3E-14	7.3	27 (22)	23	1.2
	6,6	(6,6)	5E-2	6.3E-7	4.8E-7	1.3			
			4E-2	4.8E-9	3.7E-9	1.3			
			3E-2	1.7E-11	1.3E-11	1.3			
			2E-2	1.2E-14	9.3E-15	1.3	19 (13.4)	12.8	1.5
	6,6	(4,4,4)	7E-2	-	3.6E-5	-			
			6E-2	-	3.2E-7	-			
			5E-2	6.3E-7	8.5E-9	74			
			4E-2	4.8E-9	1.4E-10	34			
			3E-2	1.7E-11	9.7E-13	18			
			2E-2	1.2E-14	7.0E-15	1.7	19 (13.4)	13.9	1.4
	6,6	(3,3,3,3)	7.5E-2	-	5.9E-5	-			
			7E-2	-	3.1E-6	-			
			6E-2	-	9.7E-8	-			
			5E-2	6.3E-7	3.2E-9	197			
			4E-2	4.8E-9	6.2E-11	78			
			3E-2	1.7E-11	4.9E-13	35			
			2E-2	1.2E-14	2.9E-14	0.4	19 (13.4)	15.4	1.2
	8,8	(8,8)	6E-2	1.2E-4	9.9E-5	1.2			
			5.5E-2	6.8E-7	5.4E-7	1.3			
			5E-2	3.5E-8	2.8E-8	1.3			
			4.5E-2	1.9E-9	1.5E-9	1.3			
			4E-2	8.1E-11	6.4E-11	1.3			
			3.5E-2	2.7E-12	2.2E-12	1.2			
			3E-2	6.6E-14	5.4E-14	1.2	19 (13.5)	12.8	1.5
	8,8	(4,4,4,4)	7.5E-2	-	3.7E-6	-			
			7E-2	-	1.8E-7	-			
			6.5E-2	-	2.3E-8	-			
			6E-2	1.2E-4	3.2E-9	37500			
			5.5E-2	6.8E-7	4.1E-10	1659			
			5E-2	3.5E-8	4.8E-11	729			
			4.5E-2	1.9E-9	4.6E-12	413			
			4E-2	8.1E-11	3.7E-13	219			
			3.5E-2	2.7E-12	5.4E-14	50			
			3E-2	6.6E-14	3.5E-14	1.9	19 (13.5)	14	1.4
	9,9	(9,9)	6E-2	4.5E-5	3.7E-5	1.2			
			5.5E-2	2.1E-7	1.7E-7	1.2			
			5E-2	8.6E-9	6.9E-9	1.2			
			4.5E-2	3.4E-10	2.7E-10	1.3			
			4E-2	1.1E-11	8.6E-12	1.3			
			3.5E-2	2.6E-13	2.1E-13	1.2	19 (13.9)	13.4	1.4
	9,9	(6,6,6)	7E-2	-	1.3E-6	-			
			6.5E-2	-	6.0E-8	-			
			6E-2	4.5E-5	5.6E-9	8393			
			5.5E-2	2.1E-7	5.1E-10	412			
			5E-2	8.6E-9	4.1E-11	210			
			4.5E-2	3.4E-10	2.6E-12	131			
			4E-2	1.1E-11	1.5E-13	73			
			3.5E-2	2.6E-13	1.3E-14	20	19 (13.9)	13.6	1.4

Table 10.4: Multistep Versus One-Step Methods : the P1 Problem.

IVP	IHO	GHF	h	Error			Time		
	p, q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
P2	8,8	(8,8)	1E-1	1.9E-5	1.6E-5	1.2			
			9E-2	4.4E-7	3.6E-7	1.2			
			8E-2	1.7E-8	1.4E-8	1.2			
			7E-2	5.7E-10	4.6E-10	1.2			
			6E-2	1.5E-11	1.2E-11	1.2	5.6 (4.1)	3.9	1.4
	8,8	(4,4,4,4)	1.4E-1	-	3.0E-6	-			
			1.3E-1	-	3.7E-7	-			
			1.2E-1	-	5.5E-8	-			
			1.1E-1	-	8.0E-9	-			
			1E-1	1.9E-5	1.1E-9	17273			
			9E-2	4.4E-7	1.3E-10	3385			
			8E-2	1.7E-8	1.7E-11	1000			
			7E-2	5.7E-10	5.5E-12	104			
			6E-2	1.5E-11	4.0E-12	3.7	5.6 (4.1)	4.9	1.1
	9,9	(9,9)	1E-1	8.5E-6	7.0E-6	1.2			
			9E-2	1.4E-7	1.1E-7	1.3			
			8E-2	3.7E-9	3.1E-9	1.2			
			7E-2	8.9E-11	7.3E-11	1.2			
			6E-2	1.6E-12	1.4E-12	1.1	7.0 (5.1)	4.8	1.5
	9,9	(6,6,6)	1.3E-1	-	3.7E-6	-			
			1.2E-1	-	2.0E-7	-			
			1.1E-1	-	1.9E-8	-			
			1E-1	8.5E-6	1.8E-9	4722			
			9E-2	1.4E-7	1.5E-10	933			
			8E-2	3.7E-9	1.1E-11	336			
			7E-2	8.9E-11	1.5E-12	59			
			6E-2	1.6E-12	1.0E-12	1.6	7.0 (5.1)	5.2	1.3

Table 10.5: Multistep Versus One-Step Methods : the P2 Problem.

IVP	IHO	GHF	h	Error			Time		
	p, q	σ		IHO	GHF	Ratio	IHO	GHF	Ratio
P3	4,4	(4,4)	5E-1	1.9E-3	1.4E-3	1.4			
			4E-1	4.0E-6	2.7E-6	1.5			
			3E-1	6.2E-8	3.9E-8	1.6			
			2E-1	3.4E-10	2.0E-10	1.7			
			1E-1	2.7E-13	9.4E-14	2.9	3.5 (2.7)	2.5	1.4
	4,4	(2,2,2,2)	6.5E-1	-	9.1E-5	-			
			6E-1	-	1.1E-5	-			
			5E-1	1.9E-3	7.0E-7	2714			
			4E-1	4.0E-6	4.3E-8	93			
			3E-1	6.2E-8	1.5E-9	43			
			2E-1	3.4E-10	1.5E-11	23			
			1E-1	2.7E-13	1.6E-14	17	3.5 (2.7)	3.3	1.1
	8,8	(8,8)	5E-1	2.6E-5	2.1E-5	1.2			
			4.5E-1	1.5E-7	1.2E-7	1.2			
			4E-1	5.4E-9	4.4E-9	1.2			
			3.5E-1	1.7E-10	1.4E-10	1.2			
			3E-1	3.7E-12	3.0E-12	1.2	3.3 (2.4)	2.2	1.5
	8,8	(4,4,4,4)	6.8E-1	-	8.9E-5	-			
			6.5E-1	-	8.3E-7	-			
			6E-1	-	4.8E-8	-			
			5.5E-1	-	6.4E-9	-			
			5E-1	2.6E-5	7.6E-10	34211			
			4.5E-1	1.5E-7	8.8E-11	1705			
			4E-1	5.4E-9	7.9E-12	684			
			3.5E-1	1.7E-10	5.4E-13	315			
			3E-1	3.7E-12	5.1E-14	73	3.3 (2.4)	2.5	1.3
	9,9	(9,9)	5E-1	1.0E-5	8.2E-6	1.2			
			4.5E-1	4.3E-8	3.5E-8	1.2			
			4E-1	1.1E-9	9.2E-10	1.2			
			3.5E-1	2.4E-11	2.0E-11	1.2			
			3E-1	3.5E-13	2.9E-13	1.2	3.9 (2.9)	2.7	1.4
	9,9	(6,6,6)	6E-1	-	6.8E-7	-			
			5.5E-1	-	2.0E-8	-			
			5E-1	1.0E-5	1.6E-9	6250			
			4.5E-1	4.3E-8	1.2E-10	358			
			4E-1	1.1E-9	7.1E-12	155			
			3.5E-1	2.4E-11	3.0E-13	80			
			3E-1	3.5E-13	1.4E-14	25	3.9 (2.9)	2.9	1.3

Table 10.6: Multistep Versus One-Step Methods : the P3 Problem.

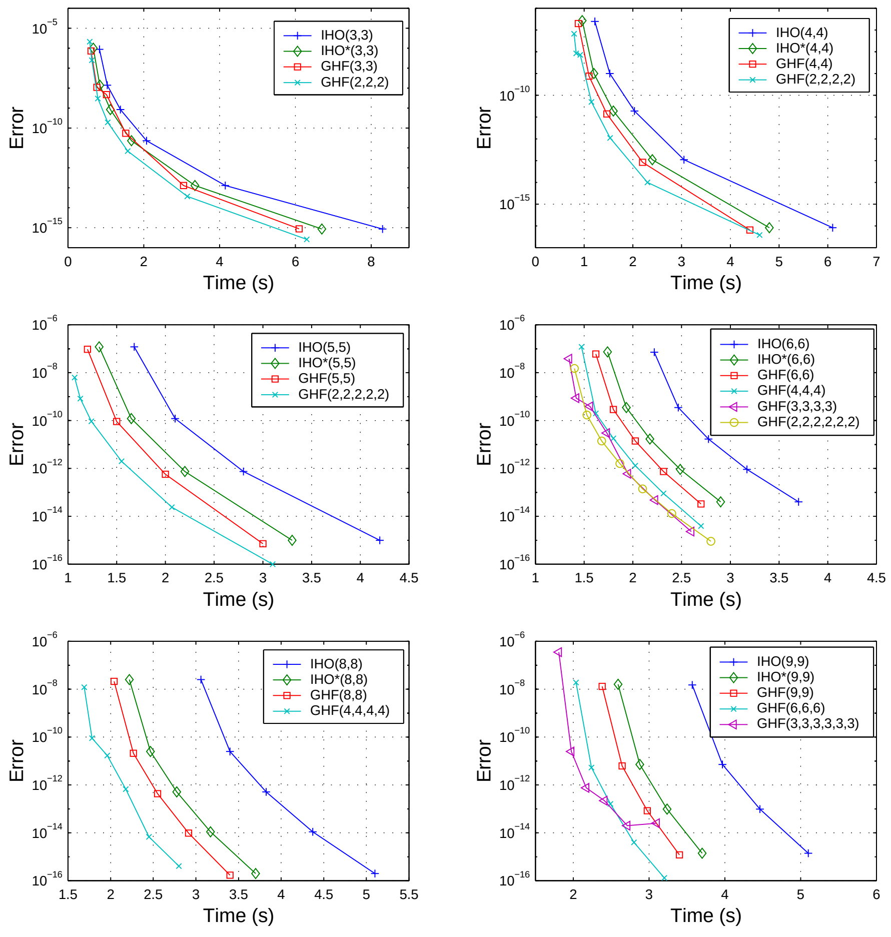

Figure 10.2: Multistep Versus One-Step Methods : the LIEN Problem.

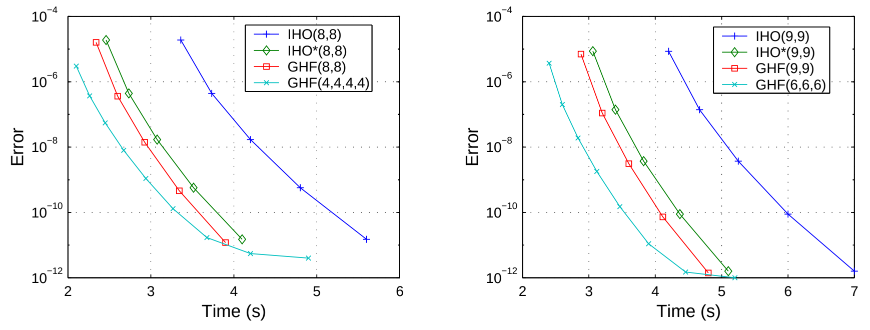

Figure 10.3: Multistep Versus One-Step Methods : the P1 Problem.

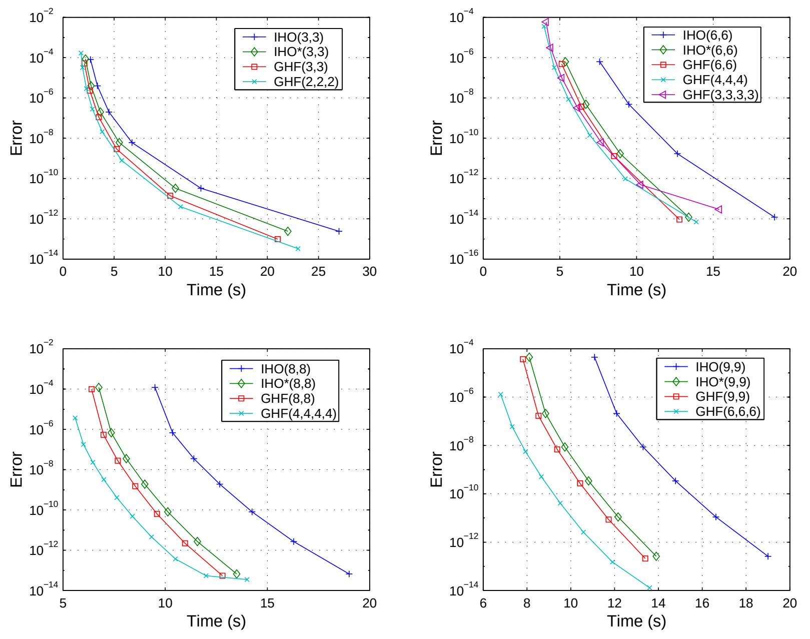

Figure 10.4: Multistep Versus One-Step Methods : the P2 Problem.

(as in standard numerical methods) but also on the corresponding formula for the truncation error. Hence, interval extensions of standard numerical methods designed for stiff problems may need smaller step sizes. Another restriction on the step size in interval methods comes from the bounding box process, whose current implementations require very small step sizes to be able to compute bounding boxes in the case of stiff problems. This explains why the differences in efficiency between interval methods are not as sharp as for traditional methods.

3. In our experiments, we always chose $\sigma_0 = \ldots = \sigma_k$. Indeed, the main cost of the method is determined by $\max_{0 \leq i \leq k}\{\sigma_i\}$ and the order of the method is maximized when $\sigma_0 = \ldots = \sigma_k$. Since the actual step sizes are sufficiently small, this choice is thus always better. If we could use larger step sizes (e.g. by improving the bounding box process), then stability requirements might make other choices preferable.

4. The results close to machine precision are not very significant since rounding errors, not the actual method, are determining the accuracy. This explains why the curves in the figures tend to join for high precisions in some cases (e.g. in LIEN, P1, P2).

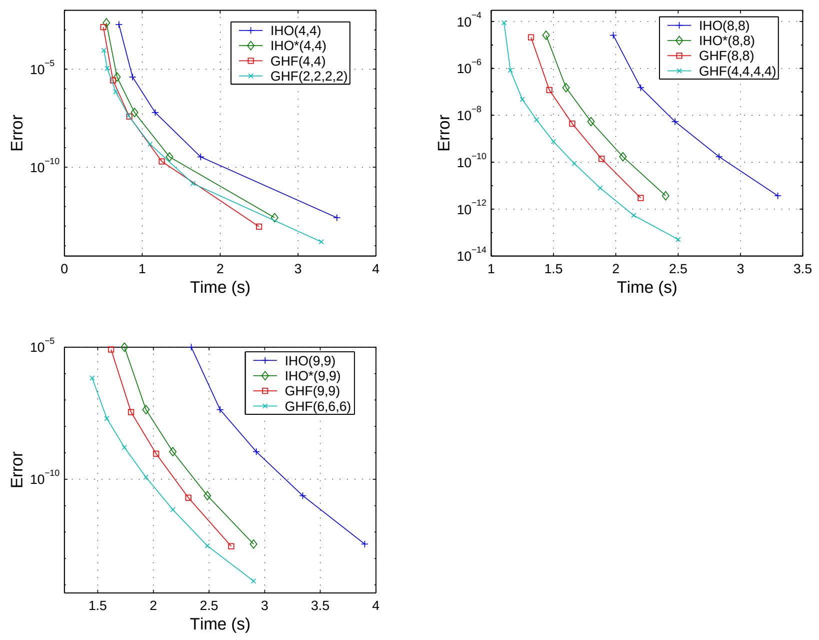

Figure 10.5: Multistep Versus One-Step Methods : the P3 Problem.

10.5 Summary

We now summarize our experimental results. The main conclusions are:

1. The one-step GHF method is almost always better than existing (one-step) interval methods;
2. When f contains few operations, the one-step GHF method outperforms multistep GHF methods (and other existing methods);
3. When f contains many operations, multistep GHF methods outperform the one-step GHF method (and other existing methods);
4. GHF methods are very versatile and can be tailored to the application at hand;
5. The experimental results confirm the theoretical analysis.

In particular, the one-step GHF method performs generally better than the IHO* method, a variant of Nedialkov's IHO method we proposed and which performed better than the original method on almost all our benchmarks. For low dimensional problems or when f contains few operations, the one-step GHF method is only slightly better than IHO*. For higher dimensional problems where f contains many operations, the one-step GHF method is asymptotically more precise (by two orders of magnitude) than IHO* for the same cost.

When f contains few operations, the one-step GHF method is more effective than multistep GHF methods which have a relatively high fixed cost. When f contains many operations, multistep GHF methods perform better than one-step methods. They may produce orders of magnitude improvements in accuracy for a given execution time. Alternatively, they may reduce computation times substantially for a given precision since they avoid expensive Jacobian computations.

Finally note that, although our implementation used a constant order and step size, it can be easily enhanced to incorporate standard order and step size control strategies, e.g., Eijgenraam's [Eij81] or Nedialkov's [Ned99] techniques.

Chapter 11

Conclusion

This work described a constraint satisfaction approach to initial value problems for parametric ordinary differential equations (i.e., ordinary differential equations where some data or initial conditions are uncertain and given by intervals).

The main novelty of the constraint satisfaction approach is to introduce, inside traditional interval methods, a pruning component which reduces the size of the predicted boxes by using relaxations of the ODE (also called filters).

The work presented an effective pruning algorithm based on

1. Relaxations of the ODE using Hermite interpolation polynomials and enclosures of their error terms;

2. Clusterings of several successive relaxations to remove variable dependency problems;

3. Traditional techniques to handle the wrapping effect;

4. Evaluation points for the relaxations that minimize their local errors.

The resulting integration algorithm was analyzed both theoretically and experimentally. The theoretical results indicate that, for the same computation costs, our algorithm provides quadratic (asymptotic) improvement in accurary over the best interval method we know of. They also show that our algorithm is significantly faster when the ODE contains many operations.

Experimental results on a variety of standard and new benchmarks validated the theoretical results. The algorithm shows significant gains in accuracy, while not degrading computational performance. The experimental results also illustrate that the approach could produce significant gain in computation time when the ODE contains many operations.

It is also important to stress the versatility of our algorithm and of our approach. On the one hand, global Hermite filters can be tailored to the problem at hand by choosing the number of interpolation points as well as the number of derivative conditions imposed at each interpolation point. On the other hand, the pruning algorithm itself is generic and new pruning techniques may easily be incorporated.

There are a wealth of topics for further research:

1. The current algorithm can be tuned in many ways, including
   - Order and step size control strategies;
   - Non-equidistant interpolation points in the Hermite filters;
   - Automatic selection of the number of interpolation points and the number of derivative conditions imposed at each interpolation point.
2. The constraint satisfaction approach is clearly in its infancy and new relaxations (e.g., based on splines, trigonometric interpolation, Legendre, Chebyshev, and Laguerre polynomials, etc.) should be investigated.
3. Compared to standard numerical methods, validated methods generally use (much) smaller step sizes and are not really suited for stiff problems. The main factors that limit the step size are
   - The need to enclose the error terms in the validated method. In our approach, it is thus important to build relaxations which minimize the enclosure of the local error;
   - The bounding box process. Current techniques to compute bounding boxes impose a severe restriction on the step size for stiff problems and are not suited for these problems. Looking for more efficient bounding box techniques is therefore a particularly important topic. For instance, it would be interesting to study how pruning techniques could help in this respect.

   If we can increase the step size by improving the techniques implementing the bounding box process, it will be very useful to analyze the stability of our approach and compare it to the stability of other validated methods. The choice of many of the parameters mentioned in 1 will be guided by stability requirements in the case of stiff problems. Furthermore, our asymptotic theory for choosing an optimal evaluation time will not necessarily be valid anymore and we will have to find new techniques for choosing a good evaluation time.
4. A possible alternative to validated methods consists of dropping the enclosures of the error terms and the bounding box process in the interval method. We thus keep the parametric aspect of the ODEs but of course the price to pay is that we loose the validated aspect of the method. However, the advantage is that larger step sizes can be used in this case. From

our experimental results, we can expect a higher gain in performance of our GHF method over the IHO$^{(*)}$ method for those larger step sizes. In addition, if we consider an ODE for which it is not possible to compute the Taylor coefficients $(u)_2, (u)_3, \ldots$ of the solution, a multistep GHF(σ) method with $\sigma_i \leq 2,\ i = 0, \ldots, k$, is the *only* current interval method which is able to integrate the ODE, since it does not need any Taylor coefficient.

5. A very promising direction of further research is the application of our approach to standard numerical methods for ODEs. Indeed, to our knowledge, the idea of evaluating a Hermite filter at a point which is different from the point at which the current value is computed is completely new. We can apply our asymptotic theory for the choice of an optimal evaluation time in the case of nonstiff problems. For stiff problems, the choice of a good evaluation time will be guided by stability requirements. Note that when $\sigma = (1, \ldots, 1)$, i.e. the Hermite interpolation polynomial reduces to a Lagrange interpolation polynomial, we can apply the classical linear stability theory to our approach.

6. Finally, it would be interesting to apply the constraint satisfaction approach to boundary value problems, where pruning arises naturally.

To conclude, we believe that the constraint satisfaction approach should be a valuable addition to existing methods for the reliable solutions of differential equations.

Appendix A

The Benchmarks

A.1 The full Brusselator (BRUS)

$$\begin{aligned}
u_1' &= 1 + u_1^2 u_2 - (u_3 + 1)u_1 \\
u_2' &= u_1 u_3 - u_1^2 u_2 \\
u_3' &= -u_1 u_3 + \alpha
\end{aligned} \tag{A.1}$$

$\alpha = 1, [t_0, t_f] = [0, 14], u(t_0) = (1, 2, 1)$

A.2 The Lorentz system (LOR)

$$\begin{aligned}
u_1' &= \sigma(u_2 - u_1) \\
u_2' &= -u_1 u_3 + \rho u_1 - u_2 \\
u_3' &= u_1 u_2 - \beta u_3
\end{aligned} \tag{A.2}$$

$\sigma = 10, \rho = 28, \beta = 8/3, [t_0, t_f] = [0, 10], u(t_0) = (15, 15, 36)$

A.3 The Two-Body problem (2BP)

$$\begin{aligned}
u_1' &= u_3 \\
u_2' &= u_4 \\
u_3' &= -\tfrac{u_1}{(u_1^2+u_2^2)^{3/2}} \\
u_4' &= -\tfrac{u_2}{(u_1^2+u_2^2)^{3/2}}
\end{aligned} \tag{A.3}$$

$[t_0, t_f] = [0, 20], u(t_0) = (1, 0, 0, 1)$

A.4 The van der Pol equation (VDP)

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= \mu(1-u_1^2)u_2 - u_1 \\ \\ \mu &= 5, [t_0,t_f] = [0,20], u(t_0) = (2,0) \end{aligned} \tag{A.4}$$

A.5 Molecular biology problem (BIO)

$$\begin{aligned} u_1' &= vi - vd\ u_3u_1/(kd+u_1) - kds\ u_1 \\ u_2' &= (v_1\ u_1/(kc+u_1))((1-u_2)/(k_1+1-u_2)) - v_2\ u_2(k_2+u_2) \\ u_3' &= u_2v_3(1-u_3)/(k_3+1-u_3) - v_4\ u_3/(k_4+u_3) \\ \\ vi &= 0.025, vd = 0.25, kd = 0.02, kds = 0.01, \\ k_1 &= 0.005, k_2 = 0.005, k_3 = 0.005, k_4 = 0.005, \\ v_1 &= 3, v_2 = 1.5, v_3 = 1, v_4 = 0.5, kc = 0.5 \\ [t_0,t_f] &= [0,3], u(t_0) = (0.01, 0.01, 0.01) \end{aligned} \tag{A.5}$$

A.6 The Oregonator (OREG)

$$\begin{aligned} u_1' &= 77.27(u_2 + u_1(1 - 8.375\times10^{-6}u_1 - u_2)) \\ u_2' &= \tfrac{1}{77.27}(u_3 - (1+u_1)u_2) \\ u_3' &= 0.161(u_1 - u_3) \\ \\ [t_0,t_f] &= [0,15], u(t_0) = (1,2,3) \end{aligned} \tag{A.6}$$

A.7 The Stiff DETEST Problem D1

$$\begin{aligned} u_1' &= 0.2(u_2 - u_1) \\ u_2' &= 10u_1 - (60 - 0.125u_3)u_2 + 0.125u_3 \\ u_3' &= 1 \\ \\ [t_0,t_f] &= [0,20], u(t_0) = (0,0,0) \end{aligned} \tag{A.7}$$

A.8 The HIRES Problem

$$\begin{aligned}
u_1' &= -1.71u_1 + 0.43u_2 + 8.32u_3 + 0.0007\\
u_2' &= 1.71u_1 - 8.75u_2\\
u_3' &= -10.03u_3 + 0.43u_4 + 0.035u_5\\
u_4' &= 8.32u_2 + 1.71u_3 - 1.12u_4\\
u_5' &= -1.745u_5 + 0.43u_6 + 0.43u_7\\
u_6' &= -280u_6u_8 + 0.69u_4 + 1.71u_5 - 0.43u_6 + 0.69u_7\\
u_7' &= 280u_6u_8 - 1.81u_7\\
u_8' &= -u_7'\\
&\\
[t_0, t_f] &= [0, 100], u(t_0) = (0, \ldots, 0, 0.0057)
\end{aligned} \tag{A.8}$$

A.9 The Lienard system (LIEN)

$$\begin{aligned}
u_1' &= u_2 - (a_1u_1 + a_2u_1^2 + \ldots + a_ru_1^r)\\
u_2' &= -u_1\\
&\\
r &= 21, a_0 = \ldots = a_r = 1, [t_0, t_f] = [0, 20], u(t_0) = (0.2, 0.1)
\end{aligned} \tag{A.9}$$

A.10 Problem P1

$$\begin{aligned}
u_1' &= -2u_2 + u_2u_3 - u_1^3\\
u_2' &= u_1 - u_1u_3 - u_2^3\\
u_3' &= u_1u_2 - u_3^3\\
&\\
[t_0, t_f] &= [0, 20], u(t_0) = (1.5, 1.2, 1)
\end{aligned} \tag{A.10}$$

A.11 Problem P2

$$\begin{aligned}
u_1' &= -u_2^3 - u_2u_3 + u_1^2u_2 - u_2^2u_1 - u_1^3\\
u_2' &= u_1u_3 - u_3^2u_1 + u_1^2u_3 - u_2^3\\
u_3' &= u_1u_2 - u_2^2u_3 + u_3^2u_2 - u_3 + u_1^2\\
&\\
[t_0, t_f] &= [0, 20], u(t_0) = (1, -1.6, 1)
\end{aligned} \tag{A.11}$$

A.12 Problem P3

$$\begin{aligned}
u_1' &= u_1^2u_2^2 - u_1^4 + \alpha u_1u_2^2 + \beta u_1^3 - u_2^4\\
u_2' &= u_1u_2^3 - u_1^3u_2 + \alpha u_2^3 + \beta u_1^2u_2\\
&\\
\alpha &= -1, \beta = 1, [t_0, t_f] = [0, 50], u(t_0) = (0.2, 0.6)
\end{aligned} \tag{A.12}$$

Appendix B

Cost of Generating Taylor Coefficients and their Jacobians

In this appendix, we count the number of arithmetic operations required in the generation of Taylor coefficients as well as their Jacobians [Moo66, Ned99]. Consider an ODE $u' = f(u)$. For simplicity, we assume that (the natural encoding of) function f contains only arithmetic operations (i.e., $+, -, *, /$). We denote by N_1 the number of $*, /$ operations in f, by N_2 the number of $\pm$ operations, and by N the sum $N_1 + N_2$. In the following, the operations $+, -, *, /$ denote either floating-point arithmetic operations (if we want to compute approximations of the Taylor coefficients and their Jacobians), or interval arithmetic operations (if we want to compute enclosures for the Taylor coefficients and their Jacobians).

B.1 Generating Taylor Coefficients

Let v and w be two functions in t. We use the notation

$$(v)_j = \frac{v^{(j)}(t_0)}{j!} \tag{B.1}$$

for the Taylor coefficients of v at the point t_0. Assume that we have already generated the Taylor coefficients $(v)_1, \ldots, (v)_j$ and $(w)_1, \ldots, (w)_j$. Then, the numbers of arithmetic operations required for the generation of the j-th Taylor coefficient respectively of the sum, subtraction, multiplication and division of v and w are given by:

$$(v \pm w)_j = (v)_j \pm (w)_j \quad : \quad 1 \text{ operation}, \tag{B.2}$$

$$(vw)_j = \sum_{l=0}^{j} (v)_l (w)_{j-l} \quad : \quad 2j+1 \text{ operations}, \tag{B.3}$$

$$(v/w)_j = (1/w)\left((v)_j - \sum_{l=1}^{j}(w)_l(v/w)_{j-l}\right) \quad : \quad 2j+1 \text{ operations.} \tag{B.4}$$

Let u be a solution of the ODE $u' = f(u)$. From (B.2-B.4), if we have already generated the Taylor coefficients $(u)_1, \ldots, (u)_{j-1}$ of the function u, then the number of arithmetic operations needed to generate its j-th Taylor coefficient $(u)_j = \frac{1}{j}(f \circ u)_{j-1}$ is given by:

$$1 + N_2 + (2j-1)N_1. \tag{B.5}$$

Thus, the total number of arithmetic operations needed to generate the Taylor coefficients $(u)_1, \ldots, (u)_j$ of the function u is equal to:

$$\begin{aligned}\sum_{l=0}^{j} 1 + N_2 + (2l-1)N_1 &= j^2N_1 + j(1+N_2) && \text{(B.6)}\\ &= j^2N_1 + O(jN_2). && \text{(B.7)}\end{aligned}$$

B.2 Generating Jacobians of Taylor Coefficients

We assume that the cost of evaluating the jacobian $\mathcal{J}(u_0)_j$ is n times the cost of evaluating the Taylor coefficient $(u_0)_j$. From (B.6)(B.7), the total number of arithmetic operations needed to generate the Jacobians $\mathcal{J}(u_0)_1, \ldots, \mathcal{J}(u_0)_j$ is thus equal to:

$$nj^2N_1 + nj(1+N_2) = nj^2N_1 + O(njN_2). \tag{B.8}$$

Appendix C

Proof of Lemma 1

C.1 Lemma 2

We first prove the following lemma.

Lemma 2 *If* $t_{k-1} < t < t_k$, *then* $\frac{\partial}{\partial v_i}\frac{\partial p_i}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) = \Theta(h^{-1})$ *for* $i = 1, \ldots, n$.

Proof Let $\Psi(t) = \mathcal{J}_v \frac{\partial p}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) - \mathcal{J} f(p(\mathbf{t}, (\mathbf{u}_0, v), t))\mathcal{J}_v p(\mathbf{t}, (\mathbf{u}_0, v), t)$. Assume that $t - t_k = \mathcal{O}(h)$. Let $t_e \in]t_{k-1}, t_k[$, $i \in 1..n$ and $q(t) = \mathcal{J}_v p_i(\mathbf{t}, (\mathbf{u}_0, v), t)$.

By definition of p, for $l = 0, \ldots, k$ and $j = 1, \ldots, \sigma_l - 1$, we have $\Psi^{(j-1)}(t_l) = 0$ and thus $h^j q_i^{(j)}(t_l) = \mathcal{O}(h)$. Furthermore, for $l = 0, \ldots, k-1$, $q_i(t_l) = 0 = \mathcal{O}(h)$ and $q_i(t_k) = \mathcal{O}(1)$.

From Proposition 5, observe that $h^j q_i^{(j)}(t) = \mathcal{O}(1)$, for $j < \sigma_s$. In particular, we have $q_i'(t_e) = \mathcal{O}(h^{-1})$ and it remains to be shown that $q_i'(t_e) = \Omega(h^{-1})$. In addition, if $h^j q_i^{(j)}(\xi_1) = \mathcal{O}(h)$ and $h^j q_i^{(j)}(\xi_2) = \mathcal{O}(h)$ with $\xi_1 < \xi_2$, then there exists $\xi_3 \in]\xi_1, \xi_2[$ such that $h^j q_i^{(j)}(\xi_3) = \Theta(1)$.

Let m_j be the minimum number of $\mathcal{O}(h)$ values of $h^j q_i^{(j)}$ at distinct *interpolation* points, and n_j the minimum number of $\mathcal{O}(h)$ values of $h^j q_i^{(j)}$ at distinct points. By continuity and by Rolle's theorem, for h sufficiently small, if $h^j q_i^{(j)}$ has at least n_j distinct $\mathcal{O}(h)$ values at $\xi_1, \ldots, \xi_{n_j}$, then $h^{j+1} q_i^{(j+1)}$ has at least $n_j - 1$ distinct zeros $\zeta_1, \ldots, \zeta_{n_j-1}$, with $\xi_1 < \zeta_1 < \xi_2 < \ldots < \xi_{n_j-1} < \zeta_{n_j-1} < \xi_{n_j}$. Thus, $\zeta_1, \ldots, \zeta_{n_j-1}$ are distinct from t_l if $\sigma_l > j+1$ (since $\sigma_l > j+1 \Rightarrow \sigma_l > j$), $l = 0, \ldots, k$. We can write:

$$\begin{aligned} n_j &= n_{j-1} - 1 + m_j \\ &= n_{j-2} - 2 + m_{j-1} + m_j \\ &= n_{j-3} - 3 + m_{j-2} + m_{j-1} + m_j \\ &\vdots \\ &= n_1 - (j-1) + \textstyle\sum_{\nu=2}^{j} m_\nu. \end{aligned} \tag{C.1}$$

Let us assume that $hq_i'(t_e) = \mathcal{O}(h)$. Since the $n_0 - 1$ distinct zeros of hq_i' arising from the n_0 distinct zeros of q_i are strictly smaller than t_{k-1}, they are distinct from $t_e > t_{k-1}$ (recall that we assume $t_0 < \ldots < t_k$). Thus, we have $n_1 = n_0 - 1 + 1 + m_1 = m_0 + m_1$. In particular, $n_{\sigma_s - 1} = m_0 + m_1 - (\sigma_s - 2) + \sum_{\nu=2}^{\sigma_s - 1} m_\nu = 2 - \sigma_s + \sum_{\nu=0}^{\sigma_s-1} m_\nu$. We can easily verify that $\sum_{\nu=0}^{\sigma_s-1} m_\nu = \sigma_s - 1$. We obtain $n_{\sigma_s - 1} = 1$. However, q_i is of degree $\sigma_s - 1$ and $q_i^{(\sigma_s - 1)}(t) \equiv c \neq 0$. Thus, $q_i^{(\sigma_s-1)}$ has no zeros, i.e. $n_{\sigma_s-1} = 0$. We have a contradiction. As a consequence, we must have $q_i'(t_e) = \Omega(h^{-1})$. □

C.2 Lemma 1

We are now in position to prove Lemma 1.

Lemma 1 *We have*

1. $\Phi(t, v) = I\lambda(t) + \mathcal{O}(1)$;
2. $\lambda(t) = \mathcal{O}(h^{-1})$;
3. $\lambda(t) = \Theta(h^{-1})$ *for* $t_{k-1} < t < t_k$.

Proof From (7.12), we have $\beta_j = \mathcal{O}(h^{-j})$ and Point 2 is straightforward. Let $i \in 1..n$ and $q(t) = \mathcal{J}_v p_i(\mathbf{t}, (\mathbf{u}_0, v), t)$. From the definition of Hermite interpolation polynomials (see Proposition 5), we can rewrite $q(t)$ and $q'(t)$ as follows

$$\begin{array}{rcl} q(t) & = & \sum_{j=0}^{\sigma_k - 1} c_j \varphi_{kj}(t) \\ & = & \sum_{j=0}^{\sigma_k-1} c_j \sum_{\nu=j}^{\sigma_k-1} \alpha_{\nu j} l_{k\nu}(t) \\ & = & \sum_{j=0}^{\sigma_k-1} d_j l_{kj}(t) \\ & = & \left(\sum_{j=0}^{\sigma_k-1} d_j \frac{(t-t_k)^j}{j!}\right) \pi(t) \\ \\ q'(t) & = & \left(\left(\sum_{j=0}^{\sigma_k-2} d_{j+1} \frac{(t-t_k)^j}{j!}\right) + \left(\sum_{j=0}^{\sigma_k-1} d_j \frac{(t-t_k)^j}{j!}\right) \sum_{\nu=0}^{k-1} \frac{\sigma_\nu}{t-t_\nu}\right) \pi(t) \end{array} \tag{C.2}$$

where $\alpha_{\nu j} \in \mathbb{R}, c_j, d_j \in \mathbb{R}^n$ are independent of t. From Proposition 5 and by definition of $q(t)$, it follows that

$$d_j = c_0 \beta_j + \mathcal{O}(h^{1-j}) = e_i \beta_j + \mathcal{O}(h^{1-j}) \tag{C.3}$$

where e_i is the i-th canonical vector. We thus have $q'(t) = e_i \lambda(t) + \mathcal{O}(1)$. By Lemma 2, $q_i'(t) = \Theta(h^{-1})$ for $t_{k-1} < t < t_k$, which proves Point 3. Let $\Phi = (\phi_1^T, \ldots, \phi_n^T)^T$. As $t - t_k = \mathcal{O}(h)$, we can write

$$\begin{array}{rcl} \phi_i(t, v) & = & q'(t) - \mathcal{J} f_i(p(\mathbf{t}, (\mathbf{u}_0, v), t) + m_e(t)) \; \mathcal{J}_v p(\mathbf{t}, (\mathbf{u}_0, v), t) \\ & = & q'(t) - \mathcal{O}(1)\mathcal{O}(1) \\ & = & e_i \lambda(t) + \mathcal{O}(1) \end{array} \tag{C.4}$$

and Point 1 is established. □

Appendix D

Evaluation of Hermite Interpolation Polynomials

When evaluating a Hermite interpolation polynomial and its derivative, an excessive propagation of rounding errors can occur if we are not careful about the way these polynomials are evaluated. This can become an important problem when the precision of the results (i.e. the size of the computed boxes) becomes close to machine precision. In this case, the precision of the results is governed by rounding errors rather than by the precision of the method itself. As a consequence, we have to find a way of evaluating Hermite interpolation polynomials and their derivatives that limits as much as possible the propagation of rounding errors.

In Section D.1, we propose a technique for evaluating the Hermite interpolation coefficients and their derivatives, and Section D.2 describes an efficient way of evaluating a Hermite interpolation polynomial with point conditions and its derivative.

D.1 Hermite Interpolation Coefficients

The Hermite interpolation coefficients $\varphi_{ij}(t)$ in Proposition 5 can be evaluated by the formula

$$\begin{array}{rcl} \varphi_{i,\sigma_i-1}(t) & = & l_{i,\sigma_i-1}(t), \;\; i=0,\ldots,k, \\ \varphi_{ij}(t) & = & l_{ij}(t) - \sum_{\nu=j+1}^{\sigma_i-1} l_{ij}^{(\nu)}(t_i)\varphi_{i\nu}(t), \;\; i=0,\ldots,k, \; j=0,\ldots,\sigma_i-2, \\ l_{ij}(t) & = & \frac{(t-t_i)^j}{j!} \prod_{\substack{\nu=0 \\ \nu\neq i}}^{k} \left(\frac{t-t_\nu}{t_i-t_\nu}\right)^{\sigma_\nu}, \;\; i=0,\ldots,k, \; j=0,\ldots,\sigma_i-1. \end{array} \tag{D.1}$$

However, this formula is not very appropriate for evaluation as rounding errors propagate quickly. We now describe a more stable technique to evaluate the coefficients $\varphi_{ij}(t)$. Formula (D.1) has to be evaluated safely in interval arithmetic. Because of rounding errors, the subexpressions involved in the computation of

$\varphi_{ij}(t)$ are intervals. Since formula (D.1) is recursive, it contains many occurences of the same variables $\varphi_{i\nu}(t)$ $(\nu > j)$, resulting in large over-approximations of $\varphi_{ij}(t)$. Hence, the idea is to factorize subexpressions in (D.1) by eliminating successively the variables $\varphi_{i,j+1}(t), \ldots, \varphi_{i,\sigma_i-1}(t)$. These variables can be eliminated by using the recursive formula (D.1). We start by eliminating the variable $\varphi_{i,j+1}(t)$ in (D.1) and by factorizing the terms involving the same variable $\varphi_{i\nu}(t),\ \nu = j+2, \ldots, \sigma_i - 1$:

$$\begin{aligned}
\varphi_{ij}(t) &= l_{ij}(t) - l_{ij}^{(j+1)}(t_i)\left(l_{i,j+1}(t) - \sum_{\nu=j+2}^{\sigma_i-1} l_{i,j+1}^{(\nu)}(t_i)\varphi_{i\nu}(t)\right) \\
&\quad - \sum_{\nu=j+2}^{\sigma_i-1} l_{ij}^{(\nu)}(t_i)\varphi_{i\nu}(t) &&\text{(D.2)} \\
&= l_{ij}(t) - l_{ij}^{(j+1)}(t_i) l_{i,j+1}(t) \\
&\quad + \sum_{\nu=j+2}^{\sigma_i-1}\left(l_{ij}^{(j+1)}(t_i) l_{i,j+1}^{(\nu)}(t_i) - l_{ij}^{(\nu)}(t_i)\right)\varphi_{i\nu}(t) &&\text{(D.3)} \\
&= \tilde{l}_{ij}(t) - \sum_{\nu=j+2}^{\sigma_i-1} \tilde{l}_{ij}^{(\nu)}(t_i)\varphi_{i\nu}(t) &&\text{(D.4)}
\end{aligned}$$

where we evaluate the expressions

$$\begin{aligned}
\tilde{l}_{ij}(t) &= l_{ij}(t) - l_{ij}^{(j+1)}(t_i) l_{i,j+1}(t), &&\text{(D.5)} \\
\tilde{l}_{ij}^{(\nu)}(t_i) &= l_{ij}^{(\nu)}(t_i) - l_{ij}^{(j+1)}(t_i) l_{i,j+1}^{(\nu)}(t_i), \quad \nu = j+2, \ldots, \sigma_i - 1. &&\text{(D.6)}
\end{aligned}$$

Starting now from (D.4), we apply the same process to eliminate successively the variables $\varphi_{i,j+2}(t), \ldots, \varphi_{i,\sigma_i-1}(t)$. The derivative coefficients $\varphi'_{ij}(t)$ can be evaluated the same way by replacing $l_{ij}(t)$ with $l'_{ij}(t)$. Because of the successive factorizations of the subexpressions, this technique is more stable than evaluation (D.1) and often produces several orders of magnitude improvements in accuracy over (D.1). Note however that the cost of the factorization technique is higher than in evaluation (D.1).

D.2 Hermite Interpolation Polynomial With Point Conditions

In this section, we discuss the propagation of rounding errors in the evaluation of a Hermite interpolation polynomial with point conditions and of its derivative, both of which are evaluated in mean-value Hermite filters (see Section 5.2). We propose an efficient evaluation technique based on a property of the Hermite interpolation coefficients.

Consider the ODE $u' = f(u)$ and let $\sigma \in \mathbb{N}^{k+1}$. The Hermite($\sigma$) interpolation polynomial wrt f and $(\mathbf{t}, \mathbf{u})$ is given by the formula (Proposition 5)

$$q(t) = \sum_{i=0}^{k} \sum_{j=0}^{\sigma_i - 1} j!(u_i)_j \varphi_{ij}(t). \tag{D.7}$$

Again, formula (D.7) and its derivative have to be evaluated safely in interval arithmetic. However, even if the evaluation of the coefficients $\varphi_{ij}(t)$ and $\varphi'_{ij}(t)$ yields very tight intervals (see Section D.1), formula (D.7) and its derivative are not always appropriate for evaluation : when the step size $h = t_k - t_0$ is small, their evaluation leads to large over-approximations of $q(t)$ and $q'(t)$. From (D.1), we can see that

$$\begin{aligned} \varphi_{ij}(t) &= \mathcal{O}(h^j), & \text{(D.8)} \\ \varphi'_{ij}(t) &= \mathcal{O}(h^{j-1}). & \text{(D.9)} \end{aligned}$$

To reduce the propagation of rounding errors in the evaluation of $q(t)$ and $q'(t)$, we first evaluate the sums of terms of the same order of magnitude, and then evaluate the sum of the results in increasing order:

$$\begin{aligned} q(t) &= r_0(t) + (r_1(t) + (\ldots + r_{\sigma_m - 1}(t))), \\ r_j(t) &= \textstyle\sum_{i \in N_j} j!(u_i)_j \varphi_{ij}(t), \quad 0 \le j < \sigma_m, \end{aligned} \tag{D.10}$$

where $\sigma_m = \max(\sigma)$ and $N_j = \{i \in 0..k \mid \sigma_i > j\}$, $0 \le j < \sigma_m$. However, formula (D.10) still suffers from instability. Observe that $q'(t) = \mathcal{O}(1)$ while $u_i \varphi'_{i0}(t) = \mathcal{O}(h^{-1})$, $0 \le i \le k$. In the derivative of formula (D.7), for small h, the terms $u_i \varphi'_{i0}(t)$ are thus much larger than the sum itself, resulting in an important loss of significant digits. We can however tackle this problem by using the following property of the Hermite interpolation coefficients.

Proposition 7 *Let $\varphi_{i0}(t)$ $(0 \le i \le k)$ be defined by Proposition 5. Then, for all $t \in \mathbb{R}$, we have $\sum_{i=0}^{k} \varphi_{i0}(t) = 1$.*

Proof From Proposition 5, we can easily verify that $(0 \le i, j \le k)$

$$\begin{aligned} \varphi_{i0}(t_j) &= \begin{cases} 1 & \text{if} \quad i = j, \\ 0 & \text{otherwise,} \end{cases} \\ \varphi_{i0}^{(\nu)}(t_j) &= 0 \quad \text{if} \quad 0 < \nu < \sigma_i. \end{aligned} \tag{D.11}$$

Let us consider the function $\psi(t) = -1 + \sum_{i=0}^{k} \varphi_{i0}(t)$. Function ψ is a polynomial of degree $\le \sigma_s - 1$, where $\sigma_s = \sum_{i=0}^{k} \sigma_i$. We have $(0 \le i \le k)$

$$\psi^{(\nu)}(t_i) = 0 \quad \text{if} \quad \sigma_i > \nu. \tag{D.12}$$

Let m_ν be the minimum number of distinct roots of $\psi^{(\nu)}$ among *interpolation* points, and n_ν the minimum number of distinct roots of $\psi^{(\nu)}$. By Rolle's theorem, if $\psi^{(\nu)}$ has at least n_ν distinct roots $\xi_1, \ldots, \xi_{n_\nu}$, then $\psi^{(\nu+1)}$ has at least $n_\nu - 1$ distinct roots $\zeta_1, \ldots, \zeta_{n_\nu - 1}$, with $\xi_1 < \zeta_1 < \xi_2 < \ldots < \xi_{n_\nu - 1} < \zeta_{n_\nu - 1} < \xi_{n_\nu}$. Thus, $\zeta_1, \ldots, \zeta_{n_\nu - 1}$ are distinct from t_i if $\sigma_i > \nu + 1$ (since $\sigma_i > \nu + 1 \Rightarrow \sigma_i > \nu$), $i = 0, \ldots, k$. We can write

$$\begin{aligned} n_\nu &= n_{\nu-1} - 1 + m_\nu \\ &= n_{\nu-2} - 2 + m_{\nu-1} + m_\nu \\ &= n_{\nu-3} - 3 + m_{\nu-2} + m_{\nu-1} + m_\nu \\ &\vdots \\ &= n_0 - \nu + \textstyle\sum_{j=1}^{\nu} m_j \\ &= -\nu + \textstyle\sum_{j=0}^{\nu} m_j. \end{aligned} \tag{D.13}$$

In particular, $n_{\sigma_s - 1} = 1 - \sigma_s + \sum_{j=0}^{\sigma_s - 1} m_j$. Since $\sum_{j=0}^{\sigma_s-1} m_j = \sigma_s$, we obtain $n_{\sigma_s - 1} = 1$. It follows that $\psi^{(\sigma_s - 1)}(t) = 0$ for all t. Since $\psi^{(\sigma_s - 1)}$ has at least one root, all derivatives $\psi^{(0)}, \ldots, \psi^{(\sigma_s - 1)}$ have at least one root. As a consequence, we have $\psi^{(0)}(t) = \ldots = \psi^{(\sigma_s - 1)}(t) = 0$ for all t and thus $\sum_{i=0}^{k} \varphi_{i0}(t) = 1$. □

By Proposition 7, we have $\sum_{i=0}^{k} \varphi'_{i0}(t) = 0$ and we can write

$$\sum_{i=0}^{k} u_i \varphi'_{i0}(t) = \sum_{i=0}^{k} (u_i - u_k + u_k)\, \varphi'_{i0}(t) \tag{D.14}$$

$$= \sum_{i=0}^{k-1} (u_i - u_k)\, \varphi'_{i0}(t). \tag{D.15}$$

It is interesting to observe that $u_i - u_k = \mathcal{O}(h)$ $(0 \le i < k)$ and the terms in sum (D.15) are now of the same order of magnitude as the sum itself. Evaluation (D.15) is thus much more stable than evaluation (D.14) and is a better choice in the evaluation of $q'(t)$. Proposition 7 can also be used to improve the evaluation of $q(t)$. We can write

$$\sum_{i=0}^{k} u_i \varphi_{i0}(t) = \sum_{i=0}^{k} (u_i - u_k + u_k)\, \varphi_{i0}(t) \tag{D.16}$$

$$= u_k + \sum_{i=0}^{k-1} (u_i - u_k)\, \varphi_{i0}(t). \tag{D.17}$$

Observe that in evaluation (D.17), the term u_k is $\mathcal{O}(1)$ while the other terms are $\mathcal{O}(h)$. This is better numerically than evaluation (D.16) where all terms are $\mathcal{O}(1)$. Finally, we obtain the following formula for evaluating $q(t)$ and $q'(t)$:

$$\begin{aligned} q(t) &= r_0(t) + (r_1(t) + (\ldots + r_{\sigma_m - 1}(t))), \\ r_j(t) &= \textstyle\sum_{i \in N_j} j!(u_i)_j \varphi_{ij}(t), \quad 1 \le j < \sigma_m, \\ r_0(t) &= u_k + \textstyle\sum_{i=0}^{k-1} (u_i - u_k)\, \varphi_{i0}(t). \end{aligned} \tag{D.18}$$

Bibliography

[AH83] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[Atk88] K. E. Atkinson. *An introduction to Numerical Analysis.* John Wiley & Sons, New York, 1988.

[BK89] H. Bauch and W. Kimmel. Solving Ordinary Initial Value Problems with Guaranteed Bounds. *Z. Angew. Math. Mech.*, 69, 1989.

[BM98] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. *Reliable Computing*, 4:361-369, 1998.

[BS96] C. Bendsten and O. Stauning. *FADBAD, a Flexible C++ Package for Automatic Differentiation Using the Forward and Backward Methods.* Technical Report 1996-x5-94, Technical University of Denmark, 1996.

[BS97] C. Bendsten and O. Stauning. *TADIFF, a Flexible C++ Package for Automatic Differentiation Using Taylor Series.* Technical Report 1997-x5-94, Technical University of Denmark, April 1997.

[CB99] J. Cruz and P. Barahona. An Interval Constraint Approach to Handle Parametric Ordinary Differential Equations for Decision Support. *Proceedings of EKBD-99*, 93-108, 1999.

[Cor88] G. F. Corliss. Applications of Differentiation Arithmetic. In R. E. Moore, editor, *Reliability in Computing*, pages 127-148. Academic Press, London, 1988.

[Cor89] G. F. Corliss. Survey of Interval Algorithms for Ordinary Differential Equations. In *Appl. Math. Comput.*, 31, 1989.

[Cor95] G. F. Corliss. Theory of Numerics in Ordinary and Partial Differential Equations (W.A. Light, M. Machetta Eds), volume Vol IV, chapter *Guaranteed Error Bounds for Ordinary Differential Equations*, pages 1-75. Oxford University Press, 1995.

[CR96] G. F. Corliss and R. Rihm. Validating an a Priori Enclosure Using High-Order Taylor Series. In *Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 228-238, 1996.

[DS76] D. P. Davey and N. F. Stewart. Guaranteed Error Bounds for the Initial Value Problem Using Polytope Arithmetic. *BIT*, 16:257-268, 1976.

[DJVH98] Y. Deville, M. Janssen, and P. Van Hentenryck. Consistency Techniques in Ordinary Differential Equations. In *CP'98*, Pisa, Italy, October 1998.

[DJVH98a] Y. Deville, M. Janssen, and P. Van Hentenryck. Consistency Techniques in Ordinary Differential Equations. To appear in *Constraints* (Special Issue on CP'98).

[Eij81] P. Eijgenraam. *The Solution of Initial Value Problems Using Interval Arithmetic.* Mathematical Centre Tracts No. 144. Stichting Mathematisch Centrum, Amsterdam, 1981.

[Enr75] W. H. Enright, T. E. Hull, and B. Lindberg. Comparing Numerical Methods for Stiff Systems of ODEs. *BIT*, 15:10-48, 1975.

[GS88] T. Gambill and R. Skeel. Logarithmic Reduction of the Wrapping Effect with Application to Ordinary Differential Equations. *SIAM J. Numer. Anal.*, 25, 1988.

[HNW87] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer-Verlag, Berlin, 1987.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[Har64] P. Hartman. *Ordinary Differential Equations.* Wiley, New york, 1964.

[Hen62] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley & Sons, New York, 1962.

[Jac75] L. W. Jackson. Interval Arithmetic Error-Bounding Algorithms. *SIAM Journal on Numerical Analysis*, 12(2):223:238, 1975.

[JDVH99] M. Janssen, Y. Deville, and P. Van Hentenryck. Multistep Filtering Operators for Ordinary Differential Equations. In *CP'99*, Alexandria, VA, October 1999.

[JVHD01] M. Janssen, P. Van Hentenryck, and Y. Deville. A Constraint Satisfaction Approach to Parametric Differential Equations In *Joint International Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, WA, August 2001.

[JVHD01a] M. Janssen, P. Van Hentenryck, and Y. Deville. Optimal Pruning in Parametric Differential Equations. Submitted to *CP'01*, Paphos, Cyprus, November 2001.

[JVHD01b] M. Janssen, P. Van Hentenryck, and Y. Deville. A Constraint Satisfaction Approach for Enclosing Solutions to Parametric Ordinary Differential Equations. Technical Report, CS-01-05, Brown University, July 2001. Submitted to *SIAM Journal on Numerical Analysis.*

[KC96] D. Kincaid and W. Cheney. *Numerical Analysis*, Second Edition, Brooks/Cole, 1996.

[Knu94] O. Knüppel. PROFIL/BIAS - A Fast Interval Library. *Computing*, 53(3-4), pp. 277-287, 1994.

[Kru69] F. Krückeberg. Ordinary Differential Equations. In E. Hansen, editor, *Topics in Interval Analysis*, page 91-97. Clarendon Press, Oxford, 1969.

[Kuh98] W. Kühn. Rigorously Computed Orbits of Dynamical Systems Without the Wrapping Effect. *Computing*, 61, No.1, 47-67, 1998.

[Kuh98a] W. Kühn. Zonotope Dynamics in Numerical Quality Control. In Hege, H-Ch. (ed.) et al., *Mathematical Visualization. Algorithms, Applications, and Numerics.* International Workshop Visualization and Mathematics, Berlin, Germany, September 16-19, 1997. Springer, Berlin, 125-134, 1998.

[Kuh99] W. Kühn. Towards an Optimal Control of the Wrapping Effect. In Csendes, Tibor (ed.), *Developments in Reliable Computing.* SCAN-98 Conference, 8th International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics. Budapest, Hungary, September 22-25, 1998. Kluwer Academic Publishers, Dordrecht, 43-51, 1999.

[Lam91] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems*, Wiley, New York, 1991.

[Loh87] R. J. Lohner. Enclosing the Solutions of Ordinary Initial and Boundary Value Problems. In *Computer Arithmetic: Scientific Computation and Programming Languages*, Wiley, 1987.

[Loh88] R. J. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen.* PhD Thesis, Universität Karlsruhe, 1988.

[Loh95] R. J. Lohner. Step Size and Order Control in the Verified Solution of IVP with ODE's. *SciCADE'95 International Conference on Scientific Computation and Differential Equations*, Stanford, California, March 28 - April 1, 1995.

[Loh01] R. J. Lohner. On the Ubiquity of the Wrapping Effect in the Computation of Error Bounds. In Kulisch, Lohner, Facius (eds.), *Perspectives on Enclosure Methods*, Springer-Verlag, Wien, 2001.

[Moo66] R.E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, N.J., 1966.

[Moo79] R.E. Moore. *Methods and Applications of Interval Analysis.* SIAM Publ., 1979.

[Ned99] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation.* Ph.D. Thesis, Computer Science Dept, Univ. of Toronto, 1999.

[NJ99] N.S. Nedialkov and K.R. Jackson. An Interval Hermite-Obreschkoff Method for Computing Rigorous Bounds on the Solution of an Initial Value Problem for an ODE, *Developments in Reliable Computing*, Kluwer, 1999.

[NJC99] N. S. Nedialkov, K. R. Jackson and G. F. Corliss. Validated Solutions of Initial Value Problems for Ordinary Differential Equations. *Applied Mathematics and Computation*, 105, pp. 21-68, 1999.

[NJ00] N.S. Nedialkov and K.R. Jackson. A New Perspective on the Wrapping Effect in Interval Methods for Initial Value Problems for Ordinary Differential Equations. In Kulisch, Lohner, Facius (eds.), *Perspectives on Enclosure Methods*, Springer-Verlag, Wien, 2001, pp. 219-264.

[Neu90] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge, 1990.

[Neu93] A. Neumaier. The Wrapping Effect, Ellipsoidal Arithmetic, Stability and Confidence Regions, *Computing*, Suppl. 9, 175-190, 1993.

[Nic78] K. Nickel. Ein Zusammenhang zwischen Aufgaben monotoner Art und Intervall-Mathematik. *Numerical Treatment of Differential Equations*, Proc. of a conf. held at Oberwolfach, July 4-10, 1976. Ed. by R. Bulirsch, R. D. Grigorieff, and J. Schröder, Springer-Verlag, Berlin, Heidelberg, New York, 121-132, 1978.

[Nic85] K. Nickel. How to Fight the Wrapping Effect. In K. Nickel, editor, *Interval Analysis 1985*, Lecture Notes in Computer Science No. 212, pages 121-132, Springer, Berlin, 1985.

[Nic86] K. Nickel. Using Interval Methods for the Numerical Solution of ODE's. *Z. angew. Math. Mech.*, 66:513-523, 1986.

[Per00] L. Perko. *Differential Equations and Dynamical Systems.* Springer-Verlag, New York, 2000.

[Ral80] L. B. Rall. Applications of Software for Automatic Differentiation in Numerical Computation. In G. Alefeld and R. D. Grigorieff, editors, Fundamentals of Numerical Computation (Computer Oriented Numerical Analysis). *Computing*, Supplement No. 2, pages 141-156, Springer-Verlag, Berlin 1980.

[Ral81] L. B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1981.

[Rih94] R. Rihm. On a Class of Enclosure Methods for Initial Value Problems. *Computing*, 53 (1994), pp. 369-377.

[Rih98] R. Rihm. Implicit Methods for Enclosing Solutions of ODEs. *J. of Universal Computer Science*, 4(2), 1998.

[Rum92] S. M. Rump. On the Solution of Interval Linear Systems. *Computing*, 47, pp. 337-353, 1992.

[Rum93] S. M. Rump. Validated Solution of Large Linear Systems. In R. Albrecht, G. Alefeld, and H. J. Stetter, editors, *Computing Supplementum*, volume 9, pages 191-212. Springer, 1993.

[Sha93] L. F. Shampine. *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, London, 1993.

[Stet90] H. J. Stetter. Validated Solution of Initial Value Problems for ODE. In C. Ullrich, ed., *Computer Arithmetic and Self-Validating Numerical Methods*, Academic Press, New York, 1990.

[Stew71] N. F. Stewart. A Heuristic to Reduce the Wrapping Effect in the Numerical Solution of $x' = f(t, x)$. *BIT*, 11:328-337, 1971.

[SB80] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 1980.

[Tsa93] E. Tsang. *Foundations of Constraint Satisfaction*, Academic Press, London and San Diego, 1993.

[VHMK97] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM Journal on Numerical Analysis*, 34(2), April 1997.

[VHMD97] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: a Modeling Language for Global Optimization.* The MIT Press, Cambridge, Mass., 1997.

[VH98] P. Van Hentenryck. A Gentle Introduction to Numerica. *Artificial Intelligence*, 103(1-2): 209-235, 1998.