A Constraint Satisfaction Approach to Parametric Differential Equations

M. Janssen (UCL), P. Van Hentenryck (Brown University), Y. Deville (UCL)

Abstract

Parametric ordinary differential equations arise in many areas of science and engineering. Since some of the data is uncertain and given by intervals, traditional numerical methods do not apply. Interval methods provide a way to approach these problems but they often suffer from a loss in precision and high computation costs. This paper presents a constraint satisfaction approach that enhances interval methods with a pruning step based on a global relaxation of the problem. Theoretical and experimental evaluations show that the approach produces significant improvements in accurracy and/or efficiency over the best interval methods.

1 Introduction

Ordinary differential equations 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* (ODE) $\mathcal{O}$ is a system of the form

$$\begin{array}{rcl} {u_1}'(t) & = & f_1(t, u_1(t), \ldots, u_n(t)) \\ & \ldots & \\ {u_n}'(t) & = & f_n(t, u_1(t), \ldots, u_n(t)) \end{array}$$

often denoted in vector notation by $u'(t) = f(t, u(t))$ or $u' = f(t, u)$. In addition, in practice, it is often the case that the parameters and/or the initial values are not known with certainty but are given as intervals. Hence traditional methods do not apply to the resulting parametric ordinary differential equations since they would have to solve infinitely many systems. Interval methods, pioneered by Moore [Moo79], provide an approach to tackle parametric ODEs as they return reliable enclosures of the exact solution at different points in time. Interval methods typically apply a one-step Taylor interval method and make extensive use of automatic differentiation to obtain the Taylor coefficients [Moo79]. Their major problem however is the explosion of the size of the boxes at successive points as they often accumulate errors from point to point and lose accuracy by enclosing the solution by a box (this is called the *wrapping effect*). Lohner's AWA system [Loh87] was an important step in interval methods which features coordinate transformation to tackle the wrapping effect. More recently, Nedialkov and Jackson's IHO method [NJ99] improved on AWA by extending an Hermite-Obreschkoff's approach (which can be viewed as a generalized Taylor method) to intervals. Note that interval methods inherently accommodate uncertain data. Hence, in this paper, we talk about ODEs to denote both traditional and parametric ODEs.

This research takes a constraint satisfaction approach to ODEs. Its basic idea [DJVH98; JDVH99] is to view the solving of ODEs as the iteration of two steps: a forward process that produces an initial enclosure of the solution at a given time (given enclosures at previous times) and a pruning process that tightens this first enclosure. The forward process is of course standard. The real novelty is the pruning component. Pruning in ODEs however generates significant challenges since ODEs contain unknown functions.

The main contribution of this paper is to show that an effective pruning technique can be derived from a relaxation of the ODE, importing a fundamental principle from constraint satisfaction into the field of differential equations. Three main steps are necessary to derive an effective pruning algorithm. The first step consists in obtaining a relaxation of the ODE by safely approximating its solution using generalized Hermite interpolation polynomials. The second step consists in using the mean-value form of this relaxation to solve the relaxation accurately and efficiently. 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. The third fundamental step 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.

Theoretical and experimental results show the benefits of the approach. The theoretical results show that the pruning step can always be implemented to make our approach faster than existing methods. In addition, it shows that our approach should be significantly faster when the function f is very complex. Experimental results confirm the theory. They show that the approach often produces many orders of magnitude improvements in accuracy over existing methods while not increasing computation times. Alternatively, at similar accuracy, other approaches are significantly slower. Of particular interest is the fact that our approach is not dependent on high-order Taylor coefficients contrary to other methods.

The rest of the paper is organized as follows. Section 2 introduces the main definitions and notations. Sections 3, 4, and 5 describe the pruning component. Sections 7 and 8 present the theoretical and experimental analyses.

2 Background and Definitions

Basic Notational Conventions Small letters denote real values, vectors and functions of real values. Capital letters denote real matrices, sets, intervals, vectors and functions of intervals. $\mathbb{IR}$ denotes the set of all *closed* intervals $\subseteq \mathbb{R}$ whose bounds are floating-point numbers. A vector of intervals $D \in \mathbb{IR}^n$ is called a *box*. If $r \in \mathbb{R}$, then $\overline{r}$ denotes the smallest interval $I \in \mathbb{IR}$ such that $r \in I$. If $r \in \mathbb{R}^n$, then $\overline{r} = (\overline{r_1}, \ldots, \overline{r_n})$. In this paper, we often use r instead of $\overline{r}$ for simplicity. If $A \subseteq \mathbb{R}^n$, then $\Box A$ denotes the smallest box $D \in \mathbb{IR}^n$ such that $A \subseteq D$. We also assume that $t_0, \ldots, t_k$, t_e and t are reals, $u_0, \ldots, u_k$ are in $\mathbb{R}^n$, and $D_0, \ldots, D_k$ are in $\mathbb{IR}^n$. Finally, we use $\mathbf{t}_k$ to denote $\langle t_0, \ldots, t_k\rangle$, $\mathbf{u}_k$ to denote $\langle u_0, \ldots, u_k\rangle$ and $\mathbf{D}_k$ to denote $\langle D_0, \ldots, D_k\rangle$.

We restrict attention to ODEs that have a unique solution for a given initial value. Techniques to verify this hypothesis numerically are well-known [Moo79; DJVH98]. Moreover, in practice, the objective is to produce (an approximation of) the values of the solution of $\mathcal{O}$ at different points $t_0, t_1, \ldots, t_m$. This motivates the following definition of solutions and its generalization to multistep solutions.

Definition 1 (Solution of an ODE) *The* solution *of an ODE $\mathcal{O}$ is the function $s(t_0, u_0)(t) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R} \to \mathbb{R}^n$ satisfying $\mathcal{O}$ for an initial condition $s(t_0, u_0)(t_0) = u_0$.*

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

$$ms(\mathbf{t}_k, \mathbf{u}_k)(t) = s(t_0, u_0)(t) \text{ if } u_i = s(t_0, u_0)(t_i) \ (1 \le i \le k)$$

where s is the solution of $\mathcal{O}$ and is undefined otherwise.

Since multistep solutions are partial functions, we generalize the definition of interval extensions to partial functions.

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

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

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

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.

Definition 4 (Bounding box) *Let $\mathcal{O}$ be an ODE system, ms be the multistep solution of $\mathcal{O}$, and $\{t_0, \ldots, t_k\} \subseteq T \in \mathbb{IR}$. A box B is a* bounding box *of ms over T wrt $\mathbf{D}_k$ and $\mathbf{t}_k$ if, for all $t \in T$, $ms(\mathbf{t}_k, \mathbf{D}_k)(t) \subseteq B$.*

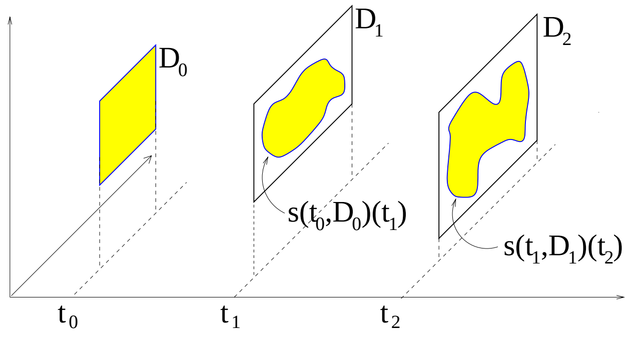

Figure 1: Successive Integration Steps

3 The Constraint Satisfaction Approach

The constraint satisfaction approach followed in this paper was first presented in [DJVH98]. It consists of a generic algorithm for ODEs that iterates two processes: (1) a *forward* process that computes initial enclosures at given times from enclosures at previous times and bounding boxes and (2) a *pruning* process that reduces the initial enclosures without removing solutions. The forward process also provides numerical proofs of existence and unicity of the solution. The intuition of the successive integration steps is illustrated in Figure 1. Forward components are standard in interval methods for ODEs. This paper thus focuses on the pruning process, the main novelty of the approach. *Our pruning component is based on relaxations of the ODE as described in the next section.* To our knowledge, no other approach uses relaxations of the ODE to derive pruning operators and the only other approach using a pruning component [NJ99; Rih98] was developed independently.

4 Pruning

The pruning component uses *safe approximations* of the ODE to shrink the boxes produced by the forward process. To understand this idea, it is useful to contrast the constraint satisfaction to nonlinear programming [VHMD97] and to ordinary differential equations. In nonlinear programming, a constraint $c(x_1, \ldots, x_n)$ can be used almost directly for pruning the search space (i.e., the carthesian products of the intervals I_i associated with the variables x_i). It suffices to take an interval extension $C(X_1, \ldots, X_n)$ of the constraint. Now if $C(I'_1, \ldots, I'_n)$ does not hold, it follows, by definition of interval extensions, that no solution of c lies in $I'_1 \times \ldots \times I'_n$. 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 $\forall\, t : u' = f(t, u)$, functions *u* and *u'* are, of course, unknown. Hence it is not obvious how to obtain a filter to prune boxes.

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 in terms of its multistep solution *ms* to obtain

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

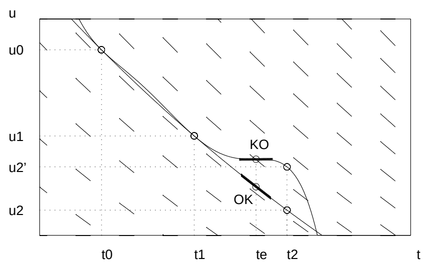

Figure 2: Geometric Intuition of the Multistep Filter

Let us denote this formula $\forall\, t : fl(\mathbf{t}_k, \mathbf{u}_k)(t)$. 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 Section 6 how to obtain interval extensions of *ms* and $\frac{\partial ms}{\partial t}$ by using 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 forward process. Once these interval extensions are available, it is possible to obtain an interval formula of the form

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

which approximates the original ODE. The above formula 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 to obtain the relation

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

which produces a computable filter. Indeed, if the relation $FL(\mathbf{t}_k, \mathbf{D}_k)(t)$ does not hold for a time t, it follows that no solution of $u' = f(t, u)$ can go through boxes $D_0, \ldots, D_k$ at times $t_0, \ldots, t_k$. The following definition and proposition capture these concepts more formally.

Definition 5 (Multistep Filters) *Let $\mathcal{O}$ be an ODE and s be its solution. A* multistep filter *for $\mathcal{O}$ is an interval relation $FL : \mathbb{R}^k \times (\mathbb{IR}^n)^k \to \mathbb{R} \to Bool$ satisfying*

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

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

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

$$\forall t \in T : FL(\mathbf{t}_k, \langle D_0, \ldots, D_{k-1}, D\rangle)(t)$$

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

$$\forall\, t : \ DMS(\mathbf{t}_k, \mathbf{D}_k)(t) = F(t, MS(\mathbf{t}_k, \mathbf{D}_k)(t)).$$

In this formula, 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 formula do not contain the solution to the ODE system. Figure 2 illustrates the intuition visually. It is generated from an actual ordinary differential equation, considers only points instead of intervals, 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 paper). 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 produced by the forward process using the operator $D_k = D_k \cap \Box\{r \in D_k \mid FL(\mathbf{t}_k, \langle D_0, \ldots, D_{k-1}, r\rangle)(t_e)\}$. The following definition is a novel notion of consistency for ODEs to capture pruning of the last r boxes.

Definition 6 (Forward Consistency of Multistep Filters) *Let FL be a multistep filter. FL is forward(r)-consistent wrt $\mathbf{t}_k$, $\mathbf{D}_k$ and t if*

$$\langle D_{k-r+1}, \ldots, D_k\rangle = \Box\{\langle v_1, \ldots, v_r\rangle \in \langle D_{k-r+1}, \ldots, D_k\rangle \mid FL(\mathbf{t}_k, \langle D_0, \ldots, D_{k-r}, v_1, \ldots, v_r\rangle)(t)\}.$$

The algorithm used in our computational results enforces forward consistency of the filters we now describe.

5 Filters

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

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

There exist standard techniques to build p and to bound e as well as its derivative and the derivative error term. Section

6 reviews how they can be derived from generalized Hermite interpolation polynomials. Here we simply assume that they are available and we show how to use them to build filters.

In the previous section, we mention how natural multistep filters can be obtained by simply replacing the multistep solution and its derivative by their interval extensions to obtain

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

It is not easy however to enforce forward consistency on a natural filter since the variables may occur in complex nonlinear expressions. In addition, in ODEs, the boxes are often small which makes mean-value forms particularly interesting.

5.1 Mean-Value Filters

The mean-value theorem for a function g states that

$$\exists \xi : x \leq \xi \leq m : g(x) = g(m) + \frac{\partial g}{\partial x}(\xi)(x - m).$$

A mean-value interval extension G of g in interval I can then be defined as

$$G(X) = G(m) + DG(I)(X - m)$$

for some m in I, where DG is an interval extension of the derivative of g. In order to obtain a mean-value filter, consider the ODE

$$\frac{\partial p}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) + \frac{\partial e}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) = f(t, p(\mathbf{t}_k, \mathbf{u}_k)(t) + e(\mathbf{t}_k, \mathbf{u}_k)(t))$$

where the multistep solution has been expanded to make the approximations and the error terms explicit. We are thus interested in finding a mean-value interval extension wrt $\mathbf{u}_k$ of the function $g(\mathbf{t}_k, \mathbf{u}_k)(t)$ defined as

$$\frac{\partial p}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) + \frac{\partial e}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) - f(t, p(\mathbf{t}_k, \mathbf{u}_k)(t) + e(\mathbf{t}_k, \mathbf{u}_k)(t)).$$

However, as mentioned earlier, e is not a computable function and we cannot compute its derivative wrt $\mathbf{u}_k$. Fortunately, it is possible to construct a mean-value filter by considering the error terms as constants and taking a mean-value interval extension of the function $g^*(\mathbf{t}_k, \mathbf{u}_k, e, de)(t)$ defined as

$$\frac{\partial p}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) + de - f(t, p(\mathbf{t}_k, \mathbf{u}_k)(t) + e).$$

If G is a mean-value interval extension of g^*, then the ODE is approximated by an expression of the form

$$\forall t : G(\mathbf{t}_k, \mathbf{D}_k, E(\mathbf{t}_k, \mathbf{D}_k)(t), DE(\mathbf{t}_k, \mathbf{D}_k)(t))(t)$$

where E and DE are interval extensions of the error term e and of its derivative with respect to t. The following two definitions capture this development more formally and can be skipped in a first reading.

Definition 7 (Mean-Value Interval Extension) *Let g be a function of signature $\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$. A mean-value interval extension of g in I' wrt G and DG, denoted by MVF(g,I'), is a function of signature $\mathbb{IR} \rightarrow \mathbb{IR} \rightarrow \mathbb{IR}$ defined as*

$$MVF(g, I')(X)(Y) = G(m)(Y) + DG(I')(Y)(X - m)$$

where G of signature $\mathbb{IR} \rightarrow \mathbb{IR} \rightarrow \mathbb{IR}$ is an interval extension of g and DG is an interval extension of the derivative of g wrt its first parameter. It is easy to extend this definition to more complex signatures.

Definition 8 (Implicit Mean-Value Filter) *Let $\mathcal{O}$ be an ODE $u' = f(t, u)$ and rewrite ms, its multistep solution, as the summation of an approximation and an error term:*

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

Let P and DP be interval extensions of p and of its derivative wrt t, let E and DE be interval extensions of the error term e and of its derivative wrt t. Define functions g^* and g^+ as

$$g^*(\mathbf{t}_k, \mathbf{u}_k, e, de)(t) = \frac{\partial p}{\partial t}(\mathbf{t}_k, \mathbf{u}_k)(t) + de - f(t, p(\mathbf{t}_k, \mathbf{u}_k)(t) + e)$$
$$g^+(\mathbf{u}_k)(\mathbf{t}_k, e, de)(t) = g^*(\mathbf{t}_k, \mathbf{u}_k, e, de)(t).$$

A mean-value filter of $\mathcal{O}$ in $\mathbf{D}_k^0$ wrt E, DE is defined as

$$MVF(g^+, \mathbf{D}_k^0)(\mathbf{t}_k, \mathbf{D}_k, E(\mathbf{t}_k, \mathbf{D}_k)(t), DE(\mathbf{t}_k, \mathbf{D}_k)(t))(t).$$

The mean-value filter for a time t produces an interval constraint of the form

$$A_0 X_0 + \ldots + A_k X_k = B$$

where each X_i is a vector of n elements and A_i is an $n \times n$ matrix. Assuming that we are interested in pruning the last box D_k, we obtain a filter as

$$D_k := D_k \cap \Box\{X_k \mid A_0 D_0 + \ldots + A_{k-1} D_{k-1} + A_k X_k = B\}.$$

Our implementation in fact uses an explicit form of this pruning operator defined as

$$D_k := D_k \cap \Box\{X_k \mid X_k = A_k^{-1} B - \textstyle\sum_{i=0}^{k-1} A_k^{-1} A_i D_i\}.$$

It is simple to make this expression forward(1)-consistent since the variables in X_k have been isolated.

Definition 9 (Explicit Mean-Value Filter) With the notations of Definition 8, let $\mathcal{O}$ be an ODE $u' = f(t, u)$ and rewrite an implicit mean-value filter of $\mathcal{O}$ in $\mathbf{D}_k^0$ wrt E, DE as

$$A_0(t) X_0 + \ldots + A_k(t) X_k = B(t)$$

where $A_i(t)$ is of signature $\mathbb{R} \rightarrow \mathbb{R}^{n \times n}$. An explicit mean-value filter of $\mathcal{O}$ in $\mathbf{D}_k^0$ wrt E, DE is given by

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

5.2 Global Filters

The explicit mean-value filter produces significant pruning of the boxes but it still suffers from a dependency problem typical of interval methods. Consider the application of the filter at two successive time steps of the method. We obtain expressions of the form

$$\begin{array}{lcl} X_k & = & B^1 - M_0^1 X_0 + \ldots + M_{k-1}^1 X_{k-1} \\ X_{k+1} & = & B^2 - M_1^2 X_1 + \ldots + M_k^2 X_k \end{array}$$

Note that the second expression considers all tuples of the form $\langle v_1, \ldots, v_k \rangle \in \langle X_1, \ldots, X_k \rangle$ to compute X_{k+1}. But only a subset of these tuples satisfy the first equation for a given $v_0 \in X_0$. The key idea to remove this dependency problem is to use a global filter which clusters a sequence of k mean-value filters together. Figure 3 gives the intuition for $k = 3$. A global filter is then obtained by predicting and pruning k boxes simultaneously. More precisely, the idea is to cluster the k successive filters

$$\begin{array}{lcl} X_k & = & B^1 - M_0^1 X_0 + \ldots + M_{k-1}^1 X_{k-1} \\ \ldots & & \\ X_{2k-1} & = & B^k - M_k^{k-1} X_{k-1} + \ldots + M_{2(k-1)}^k X_{2(k-1)} \end{array}$$

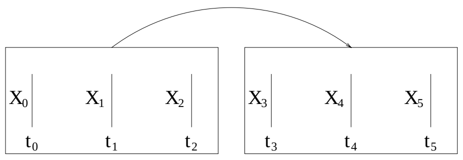

Figure 3: Intuition of the Global Filter

This filter can be turned into an explicit form by substituting X_k in the second equation, X_k and X_{k-1} in the third equation and so on. The resulting system can be written as a system

$$(X_k \dots X_{2k-1})^T = B - M(X_1 \dots X_{k-1})^T$$

where M is a square matrix of size $k \times n$. *It is important to mention such a global filter is forward(k)-consistent wrt $\mathbf{t}_{2k-1}, \mathbf{D}_{2k-1}$ and t, since the variables $X_k \dots X_{2k-1}$ have been isolated*. The dependency problem has thus been reduced to the well-known *wrapping effect* which has standard solutions in interval methods.

5.3 The Wrapping Effect

Approximating the set of solutions to an equation by a box may induce a significant loss of precision. This problem is known as the *wrapping effect* and its standard solution is to choose an appropriate coordinate system to represent the solutions compactly. Let us review briefly how to integrate this idea with our proposals. Consider a global filter that we write as $Z_1 = A_0 Z_0 + B_0$. The idea is to introduce a coordinate transformation $Y_1 = M_1 Z_1$ to obtain a system $M_1 Z_1 = M_1 A_0 Z_0 + M_1 B_0$ where $M_1 A_0$ is diagonally dominant, i.e., the non-diagonal elements are very small compared to the diagonal. If $M_1 A_0$ is diagonally dominant, then the wrapping effect induces no or little loss of precision in computing Y_i. Our pruning step can easily accommodate this idea. Iteration i defines $Y_i = M_i Z_i$ and it sends Z_i, Y_i and M_i to iteration $i+1$ of the main algorithm. Iteration $i+1$ will have to solve the system

$$Z_{i+1} = (A_{i+1} M_i^{-1}) Y_i + B_{i+1}.$$

A coordinate transformation is applied once again to obtain

$$M_{i+1} Z_{i+1} = (M_{i+1} A_{i+1} M_i^{-1}) Y_i + M_{i+1} B_{i+1}$$

so that $M_{i+1} A_{i+1} M_i^{-1}$ is diagonally dominant. The matrices M_i are computed using QR-factorizations [Loh87].

6 Hermite Filters

We now show how to build interval extensions of p, $\partial p/\partial t$, e, $\partial e/\partial t$ by using Hermite interpolation polynomials. Informally, a Hermite interpolation polynomial approximates a continuously differentiable function f and is specified by imposing that its values and the values of its successive derivatives at various points be equal to the values of f and of its derivatives at the same points. Note that the number of conditions at the different points may vary.

Definition 10 (Hermite(σ) Interpolation Polynomial) *Let $t_i \in \mathbb{R}, u_i^{(0)} = u_i \in \mathbb{R}^n$ and $u_i^{(j)} = f^{(j-1)}(t_i, u_i),\ i = 0, \dots, k,\ j = 0, \dots, \sigma_i$. Consider the ODE $u' = f(t,u)$. Let $\sigma \in \mathbb{N}^{k+1}, \sigma_i \neq 0,\ i = 0, \dots, k,\ \sigma_s = \sum_{i=0}^{k} \sigma_i$. The Hermite($\sigma$) interpolation polynomial wrt f and $(\mathbf{t}_k, \mathbf{u}_k)$, is the unique polynomial q of degree $\leq \sigma_s - 1$ satisfying*

$$q^{(j)}(t_i) = u_i^{(j)}, \quad j = 0, \dots, \sigma_i - 1,\ i = 0, \dots, k.$$

It is easy to take interval extensions of a Hermite interpolation polynomial and of its derivatives. The only remaining issue is to bound the error terms. The following standard theorem (e.g., [SB80]) provides the necessary theoretical basis.

Theorem 1 (Hermite Error Term) *Let $p(\mathbf{t}_k, \mathbf{u}_k, \bullet)$ be the Hermite(σ) interpolation polynomial wrt f and $(\mathbf{t}_k, \mathbf{u}_k)$. Let $u(t) \equiv ms(\mathbf{t}_k, \mathbf{u}_k, t)$, $T = \Box\{t_0, \dots, t_k, t\}$, $\sigma_s = \sum_{i=0}^{k} \sigma_i$ and $w(t) = \prod_{i=0}^{k} (t - t_i)^{\sigma_i}$. Then, for $i = 1, \dots, n$,*
- *$\exists\, \xi_i \in T : e_i(\mathbf{t}_k, \mathbf{u}_k, t) = \frac{1}{\sigma_s!} f_i^{(\sigma_s - 1)}(\xi_i, u(\xi_i)) w(t)$;*
- *$\exists\, \xi_{1,i}, \xi_{2,i} \in T : \frac{\partial e_i}{\partial t}(\mathbf{t}_k, \mathbf{u}_k, t) =$ $\frac{1}{\sigma_s!} f_i^{(\sigma_s - 1)}(\xi_{1,i}, u(\xi_{1,i})) w'(t) + \frac{1}{(\sigma_s+1)!} f_i^{(\sigma_s)}(\xi_{2,i}, u(\xi_{2,i})) w(t)$.*

How to use this theorem to bound the error terms? It suffices to take interval extensions of the formula given in the theorem and to replace $\xi_i, \xi_{1,i}, \xi_{2,i}$ by T and $u(\xi_i), u(\xi_{1,i}), u(\xi_{2,i})$ by a bounding box for the ODE over T. In the following, we call *Hermite(σ) filters*, filters based on Hermite(σ) interpolation and we denote a global Hermite(σ) filter by GHF(σ). The following novel result is fundamental and specifies the speed of convergence of Hermite filters. It shows that the order of natural and mean-value Hermite(σ) filters is the sum of the elements in σ, i.e., the number of conditions imposed on the interpolation.

Proposition 2 (Order) *Let FL be a natural or mean-value Hermite(σ) filter for ODE $u' = f(t,u)$. Let $t_e - t_k = \Theta(h)$, $t_{i+1} - t_i = \Theta(h),\ i = 0, \dots, k-1,\ \sigma_s = \sum_{i=0}^{k} \sigma_i$ and $D_k = \Box\{u \in \mathbb{R}^n \mid FL(\mathbf{t}_k, \langle u_0, \dots, u_{k-1}, u \rangle)(t_e)\}$. Then, under some weak assumptions on f, $w(D_k) = O(h^{\sigma_s + 1})$.*

7 Theoretical Cost Analysis

We now analyse the cost of our method and compare it to Nedialkov's IHO(p,q) method [NJ99], the best interval method we know of. We use the following assumptions. At each step, the forward process uses Moore's Taylor method and the pruning component applies a global Hermite filter together with coordinate transformations (using Lohner's QR-factorization technique). For simplicity of the analysis, 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 $\partial f^{(r)}/\partial u$ is n times the cost of evaluating $f^{(r)}$. We define σ_m as $\max(\sigma)$, $\sigma_s = \sigma_0 + \dots + \sigma_k$, $p+q+1 = \sigma_s$, $q \in \{p, p+1\}$. We also report separately interval arithmetic operations involved in (1) products of a real and an interval matrix (Cost-1) and (2) the generation of Jacobians (Cost-2). Table 7 reports the main cost of a step in the IHO(p,q) method (IHO in the table) and our method GHF(σ) (GHF in the table). It also shows the complexity of two particular cases of GHF(σ). The first case (GHF-1) corresponds to a polynomial with only two interpolation points ($k = 1$, $|\sigma_1 - \sigma_0| \leq 1$), while the second case corresponds to a polynomial imposing two conditions on every interpolation points ($\sigma_0 = \dots = \sigma_k = 2$). Note that the methods are of the same order in all cases.

	Cost-1	Cost-2
IHO	$-$	$2\lfloor\frac{\sigma_s}{2}\rfloor^2 nN_1$ $+O(\sigma_s nN_2)$
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	$(\frac{7}{8}\sigma_s-\frac{21}{4})\sigma_s^2n^3$	$(\sigma_s-2)nN$

Table 1: Complexity Analysis.

IVP	GMF	IHO	h	Precision			Time	
	σ	p, q		IHO	GMF	Ratio	IHO	GMF
HILB	(2,2,2)	2,3	8E-3	1.8E-1	2.5E-3	72	0.09	0.08
			6E-3	3.8E-4	1.6E-5	24	0.12	0.10
			4E-3	1.6E-5	2.8E-7	57	0.18	0.15
			2E-3	1.5E-7	1.1E-9	136	0.35	0.31
			1E-3	2.0E-9	7.8E-12	256	0.70	0.68
BRUS	(2,2,2)	2,3	1E-1	1.1E-1	3.3E-4	333	0.56	0.55
			7.5E-2	2.2E-4	8.7E-6	25	0.74	0.74
			5E-2	1.1E-5	1.9E-7	58	1.10	1.10
			2.5E-2	1.5E-7	1.1E-9	136	2.20	2.20
BIO1	(2,2,2)	2,3	1.5E-1	8.9E-3	2.5E-4	36	0.16	0.14
			1E-1	2.8E-5	1.0E-6	28	0.23	0.20
			5E-2	2.2E-7	2.8E-9	79	0.44	0.42
			2.5E-2	2.6E-9	1.6E-11	162	0.87	0.87
2BP	(5,5)	4,5	1E-1	3.7E-4	7.2E-6	51	2.10	1.10
			7.5E-2	1.0E-6	1.7E-8	59	2.80	1.70
	(7,6)	6,6	1.25E-1	1.0E-1	3.0E-4	333	2.20	1.40
			1E-1	1.4E-6	9.2E-8	15	2.70	2.00
3BP	(2,2,2,2)	3,4	3E-2	6.9E-2	3.5E-4	197	0.55	0.45
			2E-2	2.1E-4	6.5E-7	323	0.83	0.72
			1E-2	5.3E-8	9.3E-11	570	1.60	1.60
	(5,5,5)	7,7	3E-2	2.4E-2	1.4E-4	171	1.30	0.92
			2E-2	4.4E-7	1.5E-9	293	1.90	1.40
LOR	(3,3)	2,3	1E-2	3.1E+1	4.5E-2	689	2.50	1.90
			7.5E-3	4.0E-1	4.9E-3	82	3.30	2.60
			5E-3	2.5E-2	2.7E-4	93	5.00	3.90
			2.5E-3	3.7E-4	2.0E-6	185	9.90	8.50
BIO2	(4,3)	3,3	6E-3	1.0E-3	4.4E-4	2.3	2.80	2.20
			4E-3	4.0E-6	1.2E-6	3.3	4.10	3.20
			2E-3	1.1E-8	7.3E-10	15	8.30	6.40

Table 2: Experimental Results.

The first main result is that *GHF-1 is always cheaper than IHO*, which means that our method can always be made to run faster by choosing only two interpolation points. (The next section will show that substantial improvement in accuracy is also obtained in this case). GHF-2 is more expensive than GHF-1 and IHO when f is simple. However, *when f contains many operations (which is often the case in practical applications), GHF-2 can become 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. It also shows the versatility of the approach that can be taylored to the application at hand.

8 Experimental Analysis

We now report experimental results of our C++ implementation on some standard benchmarks [HNW87; Loh87] and two molecular biology problems, which are real-life parametric ODEs given to us by a biologist:

	u_0	$[t_0, t_f]$
Hilbert quadratic problem (HILB):	(2,4)	[0,0.15]
Full Brusselator (BRUS):	(1,2,1)	[0,14]
Two-body problem (2BP):	(1,0,0,1)	[0,24]
Three-body problem(3BP):	(1.2,0,0,-1.05)	[0,1.3]
Lorentz system (LOR):	(15,15,36)	[0,10]
Molecular biology problem (BIO1):	(0.1,0.56,0.14)	[0,4]
Molecular biology problem (BIO2):	(0,0.4,0.3,0.5)	[0,3]

Table 2 compares GHF(σ) and IHO(p, q) methods of the same order. It reports the precision at the last step and execution time of both methods for the same (constant) stepsize. The experimental results follow the same assumptions as in the theoretical analysis. The forward process uses Moore's Taylor method of order $q+1$ (same order as the predictor used in IHO(p, q)) and a Taylor series method of order σ_s to compute a bounding box, except for BIO1 and BIO2 where we use a series of order 1. The choice of the evaluation time t_e involved in GHF(σ) has not been discussed yet. So far we have no theoretical result about the optimal choice of t_e. We use a simple binary search algorithm to determine a good value for t_e at the beginning of or during the integration. In our experiments, we chose t_e between the last two interpolation points, keeping the distance constant throughout the integration. Our results could be further improved by using a variable distance.

The results indicate that our method produces orders of magnitude improvements in accuracy and runs faster than the best known method. The gain in precision is particularly significant for lower orders. The theoretical results are also confirmed by the experiments. When f contains many operations (e.g. in 3BP), using many interpolation points is particularly effective. For very complex functions, the gain in computation time could become substantial. When f is simple, using few interpolation points becomes more interesting.

As a consequence, we believe that a constraint satisfaction approach to parametric ordinary differential equations is a very promising avenue that complements well traditional approaches.

References

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

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

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

[Loh87] Lohner R. J. Enclosing the solutions of ordinary initial and boundary value problems. In *Computer Arithmetic: Scientific Computation and Programming Languages*, Wiley, 1987.

[Moo79] R.E. Moore. *Methods and Applications of Interval Analysis.* SIAM Publ., 1979.

[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.

[Rih98] R. Rihm. Implicit Methods for Enclosing Solutions of ODEs. *J. of Universal Computer Science*, 4(2), 1998

[SB80] Stoer J., Bulirsch R. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 1980.

[VHMD97] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: a Modeling Language for Global Optimization.* The MIT Press, Cambridge, Mass., 1997.