SIAM J. NUMER. ANAL.
Vol. 40, No. 5, pp. 1896–1939

A CONSTRAINT SATISFACTION APPROACH FOR ENCLOSING SOLUTIONS TO PARAMETRIC ORDINARY DIFFERENTIAL EQUATIONS[*]

MICHA JANSSEN[†], PASCAL VAN HENTENRYCK[‡], AND YVES DEVILLE[†]

Abstract. This paper considers initial value problems 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.

Key words. ordinary differential equation, interval methods, constraint satisfaction polynomial

AMS subject classifications. 65L05, 65G20

PII. S0036142901392316

1. Introduction.

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 ODE $\mathbb{O}$ is a system of the form

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

often denoted in vector notation by $u'(t) = f(u(t))$ or $u' = f(u)$.[1] An IVP is an ODE with an initial condition $u(t_0) = u_0$. 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 may not be the simplest way to approach the resulting parametric ODEs since, in essence, they would have to solve infinitely many systems. *Interval methods*, pioneered by Moore [21], provide an approach to tackle parametric ODEs. They 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. In addition, they inherently accommodate uncertainty in the parameters or initial values by using intervals instead of floating-point numbers. In this paper, we talk about ODEs to denote both traditional and parametric ODEs.

[*]Received by the editors July 12, 2001; accepted for publication (in revised form) April 3, 2002; published electronically December 3, 2002. This research was partially supported by the actions de recherche concertée ARC/95/00-187 and an NSF NYI award.
http://www.siam.org/journals/sinum/40-5/39231.html

[†]UCL, 2 Place Sainte Barbe, B-1348 Louvain-La-Neuve, Belgium (mja@info.ucl.ac.be, yde@info.ucl.ac.be).

[‡]Brown University, Box 1910, Providence, RI 02912 (pvh@cs.brown.edu).

[1]Only autonomous systems are considered in this paper, but it is not difficult to generalize the results to nonautonomous systems.

Traditional interval 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 interval 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 [27] to obtain the Taylor coefficients [8, 16, 21, 22]. However, the major problem of such methods 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 [20] 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 method [24] 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 and Makino [4] 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.

The research described in this work takes a constraint satisfaction approach to ODEs. Its basic idea [7, 13, 14] 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 nontrivial constraint from a *relaxation* of the ODE, a key concept in constraint satisfaction [32]. This constraint can then be used to prune the solution space at the various integration points.

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 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 of obtaining a relaxation of the ODE by safely approximating its solution using Hermite interpolation polynomials.
2. The second step consists of using the mean-value form of this relaxation for more accuracy and efficiency. Unfortunately, these two steps, which were sketched in [13], are not sufficient, and the resulting pruning algorithm still suffers from traditional problems of interval methods.
3. The third fundamental step [14] consists of 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 relax-

[2]Observe that interval extensions of predictor/corrector methods (e.g., [24]) can also be viewed as the composition of a predictor and a pruning step.

[3]Global constraints in ODEs have also been found useful in [6]. The problem and the techniques in [6] are, however, fundamentally different.

ation 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.

The rest of the paper is organized as follows. Section 2 introduces the main definitions and notations. Section 3 gives a high-level overview of the constraint satisfaction approach to parametric ODEs. The next four sections are the core of the paper. Section 4 introduces multistep filters, section 5 presents multistep Hermite filters as a special case of multistep filters, section 6 describes how to choose an evaluation time to minimize the local error of a multistep Hermite filter, and section 7 presents the overall algorithm. Sections 8 and 9 report the theoretical and experimental analyses, and section 10 concludes the paper.

2. Background and definitions.

2.1. 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*[4] of the inverse of M. A relation is a function $r : \mathbb{R}^n \rightarrow Bool$, 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 $\omega((I_1, \dots, I_n)) = (\omega(I_1), \dots, \omega(I_n))$ if $I_i \in \mathbb{IR}$. If $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$, $x = (x_1, \dots, x_m)$ and $\tilde{x} = (x_{i_1}, \dots, x_{i_p})$ with $i_1, \dots, i_p \in 1..m$, then $\mathcal{J}_{\tilde{x}} g(x)$ denotes the Jacobian matrix

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

In particular, we write $\mathcal{J} g(x) = \mathcal{J}_x g(x)$ (differentiation w.r.t. 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

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

values of the solution at times $t_0, \dots, t_{k-1}$ used to compute the new value at time t_k (k-step approach).

NOTATION 1 (boldface notations). *Let A be a set and $a_i \in A$, where $i \in \mathbb{N}$. We use the following boldface notations.*

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

Observe that $\mathbf{a}_0 = (a_0, \dots, a_{k-1})$, $\mathbf{a}_1 = (a_k, \dots, a_{2k-1})$, and $\mathbf{a} = (a_0, \dots, a_k)$. 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:*

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

The notations extend componentwise 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). The implementation of course uses outwardly directed rounding.

2.2. Basic definitions. As is 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.[5] 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 [25, 21, 22, 5, 23]. 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.

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*

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

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

$$s(t_0, x, t) = s(0, x, \tau),$$

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

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

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

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

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

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 lth 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 section 3), 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 $ms : A \subseteq \mathbb{R}^{k+1} \times (\mathbb{R}^n)^{k+1} \times \mathbb{R} \to \mathbb{R}^n$:*

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

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 [31]. 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*

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

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*

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

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 section 5).

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 $t \in T$, $ms(\mathbf{t}, \mathbf{D}, t) \subseteq B$.*

2.3. 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:

$$A = m(A) + s(A).$$

In this paper, the midpoint technique is used in the following two cases:

1. enclosing a set of real matrix-matrix-vector products (see sections 4.4 and 4.5);
2. converting an implicit interval linear system into an explicit one by matrix inversion (see section 4.2).

Assume that we are interested in enclosing the set

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

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 $A(BD)$. In general, this product does not yield accurate results because of the wrapping effect which occurs in the product $E = BD$ and in the product AE. Another straightforward 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 [15] 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

$$P \subseteq Q = (m(A)B)D + s(A)(BD). \tag{1}$$

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.

Now consider the implicit interval linear system

$$\begin{array}{l} A_0X_0 + A_1X_1 = B, \\ X_0 \subseteq D_0,\ X_1 \subseteq D_1, \end{array} \tag{2}$$

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

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

We are interested in converting the system (2) into a system

$$X_0 = CX_1 + E,$$

which is explicit in the variable X_0 and such that

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

A straightforward solution consists of computing an enclosure A_0^{-1} of the inverse of A_0, multipling both sides of (2) by A_0^{-1}, and rearranging the parentheses:

$$X_0 = -(A_0^{-1}A_1)X_1 + A_0^{-1}B. \tag{3}$$

However, the system (3) 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 [23]).
- We have to multiply the two interval matrices A_0^{-1} and A_1.

To eliminate these operations, we apply the midpoint technique both on A_0 and A_1 in (2). By distributivity, we have

$$m(A_0)X_0 = -m(A_1)X_1 + B - s(A_0)X_0 - s(A_1)X_1. \tag{4}$$

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 $s(A_1)$:

$$m(A_0)X_0 = -m(A_1)X_1 + B - s(A_0)D_0 - s(A_1)D_1. \tag{5}$$

Note that it is important to have precise enclosures D_1 and D_2. 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 (5) by $m(A_0)^{-1}$, and we rearrange the parentheses:[6]

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

Observe that, in this last system, there is no interval matrix inversion and no product of two interval matrices.

3. The constraint satisfaction approach.

The constraint satisfaction approach followed in this work was first presented in [7]. 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.

The intuition of the successive steps is illustrated in Figure 1. Bounding box and predictor 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, 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 [24, 28] 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^*).

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 [30, 31] and to ODEs. In nonlinear programming, a constraint $c(x_1, \ldots, x_n)$ 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 $C(X_1, \ldots, X_n)$ of the constraint. Now if $C(I'_1, \ldots, I'_n)$ does not hold, it follows, by the definition of interval extensions, that no solution of c lies in $I'_1 \times \cdots \times I'_n$. The interval extension can be seen as a filter

[6]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.

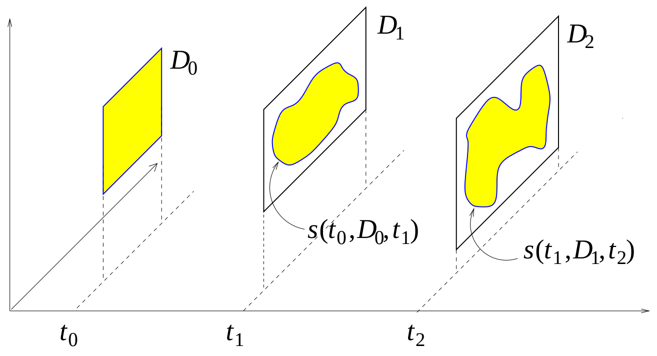

FIG. 1. *Successive integration steps.*

that can be used for pruning the search space in many ways. For instance, Numerica uses box(k)-consistency on these interval constraints [31]. ODEs raise new challenges. In an ODE for all $t: \; u' = f(u)$, functions u and u' are, of course, unknown. Hence, it is not obvious how to obtain a filter to prune boxes.

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

$$\forall\, t: \; \frac{\partial ms}{\partial t}(\mathbf{t}, \mathbf{u}, t) = f(ms(\mathbf{t}, \mathbf{u}, t)). \tag{6}$$

Let us denote this relation for all $t: fl(\mathbf{t}, \mathbf{u}, t)$. 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 section 5 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

$$\forall\, t : FL(\mathbf{t}, \mathbf{D}, t), \tag{7}$$

which approximates the original ODE *safely*; i.e., 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 $D_0, \ldots, D_k$ at times $t_0, \ldots, t_k$. Relation (7) 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

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

which produces a computable filter. The relation FL is a *relaxation* of the ODE (6) in a constraint satisfaction sense [32]; 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.

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*

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

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

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

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

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

In this relation, the left-hand side of the equation represents *the approximation of the slope of u* while the right-hand side 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 2 illustrates the intuition. It is generated from an actual ODE, 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 paper.) As can be easily seen, 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 since we assume that the values u_0 and u_1 were correct at t_0 and t_1. 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 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

$$D_k^* = \Box\{r \in D_k^- \ \mid \ FL(\mathbf{t}, (D_0^*, \ldots, D_{k-1}^*, r), t_e)\},$$

which is defined in terms of an evaluation time t_e. *One of the main results of this paper consists of showing that t_e can be chosen optimally (in an asymptotic sense) to maximize pruning.* The following definition is a novel notion of consistency for ODEs to capture pruning of the last r boxes.[7]

[7]We will give an explicit form for D_k^* later in the paper.

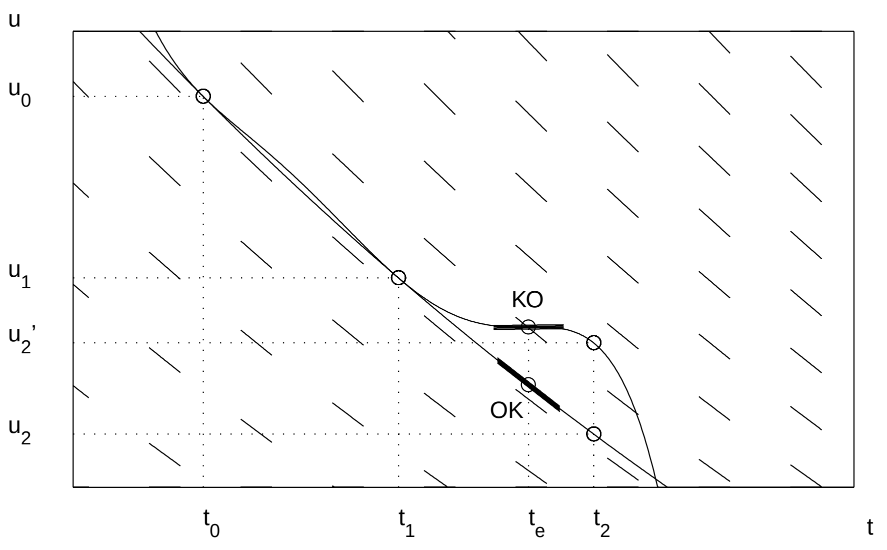

FIG. 2. *Geometric intuition of the multistep filter.*

DEFINITION 7 (backward consistency of multistep filters). *A multistep filter FL is* backward-consistent *in* $(\mathbf{t}, \mathbf{D})$ *for time e if*

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

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* $(e_0, \dots, e_{r-1})$ *if*

$$\mathbf{D}_{k..k+r-1} = \Box\{\mathbf{u}_{k..k+r-1} \in \mathbf{D}_{k..k+r-1} \mid \exists \mathbf{u}_0 \in \mathbf{D}_0 : \forall\, 0 \le i < r : FL_i(\mathbf{t}_{i..k+i}, \mathbf{u}_{i..k+i}, e_i)\}. \tag{8}$$

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

PROPOSITION 2 (property of backward consistency). *If a system of* $r+1$ $(r > 0)$ *successive multistep filters* $\{FL_i\}_{0 \le i \le r}$ *is backward*$(r+1)$*-consistent in* $(\mathbf{t}_{0..k+r}, \mathbf{D}_{0..k+r})$ *for time vector* $(e_0, \dots, e_r)$, *then the system*

1. $\{FL_i\}_{0 \le i < r}$ *is backward*(r)*-consistent in* $(\mathbf{t}_{0..k+r-1}, \mathbf{D}_{0..k+r-1})$ *for time vector* $(e_0, \dots, e_{r-1})$;
2. $\{FL_i\}_{1 \le i \le r}$ *is backward*(r)*-consistent in* $(\mathbf{t}_{1..k+r}, \mathbf{D}_{1..k+r})$ *for time vector* $(e_1, \dots, e_r)$.

In the next section, we introduce coordinate transformations in multistep filters to represent the sets of solutions compactly, i.e., to handle the wrapping effect (see section 4.5). 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* $\mathbf{a} : \mathbb{R}^{n(k+1)} \rightarrow \mathbb{R}^{n(k+1)}$ *such that*

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

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

$$\begin{aligned} \mathbf{Y}_{k..k+r-1} = \Box\{&\mathbf{y}_{k..k+r-1} \in \mathbf{Y}_{k..k+r-1} \mid \exists \mathbf{y}_0 \in \mathbf{Y}_0 : \\ &\forall 0 \le i < r : FL_i(\mathbf{t}_{i..k+i}, \mathbf{a}_{i..k+i}(\mathbf{y}_{0..k+r-1}), e_i)\}. \end{aligned} \tag{9}$$

Note that Proposition 2 also holds for generalized backward consistency. In the rest of this paper, 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 now describe.

4. Multistep filters.

Filters rely on interval extensions of the multistep solution and of its derivative w.r.t. 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.,

$$ms(\mathbf{t}, \mathbf{u}, t) = p(\mathbf{t}, \mathbf{u}, t) + e(\mathbf{t}, \mathbf{u}, t). \tag{10}$$

There exist standard techniques to build p and $\frac{\partial p}{\partial t}$ and to bound e and $\frac{\partial e}{\partial t}$. Section 5 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.

4.1. Natural filters.

Section 3 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

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

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.

4.2. Mean-value filters.

Mean-value forms. 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. MVFs 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 so-called *wrapping effect*, a crucial problem in interval methods for ODEs to be discussed in sections 4.3 and 4.5. As a consequence, MVFs 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.

Implicit mean-value filters. Consider the function

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

If the multistep solution ms is defined at $(\mathbf{t},\mathbf{u})$, i.e., the ODE has a solution going through $u_0,\dots,u_k$ at $t_0,\dots,t_k$, then, by (10), we have the relation

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

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 $(1 \le i \le n)$

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

where

$$\begin{aligned}&\phi_i(\mathbf{t},\mu_i,\xi_i,t) = \mathcal{J}_{\mathbf{u}}\tfrac{\partial p_i}{\partial t}(\mathbf{t},\mu_i,t) - \mathcal{J}f_i(p(\mathbf{t},\mu_i,t)+\xi_i)\mathcal{J}_{\mathbf{u}}p(\mathbf{t},\mu_i,t),\\ &\psi_i(\mathbf{t},\mu_i,\xi_i,t) = \mathcal{J}f_i(p(\mathbf{t},\mu_i,t)+\xi_i)\end{aligned}$$

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*

(11)
$$\begin{gathered}FL(\mathbf{t},\mathbf{D},t) \Leftrightarrow\\ \delta(\mathbf{t},\mathbf{m}^0,m_e,m_{de},t) + \Delta(\mathbf{t},\mathbf{D}^0,E(\mathbf{t},\mathbf{D}^0,t),DE(\mathbf{t},\mathbf{D}^0,t),t)\ (\mathbf{X},E_m,DE_m) = 0,\end{gathered}$$

where

(12)
$$\begin{aligned}&\Delta \text{ is an interval extension of the function } \mathcal{J}_{(\mathbf{u},e,de)}\delta,\\ &E \text{ and } DE \text{ are interval extensions, respectively, of } e \text{ and } \tfrac{\partial e}{\partial t},\\ &\mathbf{D} \subseteq \mathbf{D}^0,\\ &\mathbf{X} = \mathbf{D} - \mathbf{m}^0, E_m = E(\mathbf{t},\mathbf{D}^0,t) - m_e, DE_m = DE(\mathbf{t},\mathbf{D}^0,t) - m_{de},\\ &\mathbf{m}^0 = m(\mathbf{D}^0), m_e = m(E(\mathbf{t},\mathbf{D}^0,t)), m_{de} = m(DE(\mathbf{t},\mathbf{D}^0,t)).\end{aligned}$$

Formula (11) is called implicit because $\mathbf{D}$ appears implicitly. The Jacobians in (12) can be computed by means of automatic differentiation tools (see, e.g., [27]). 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 $\mathcal{O}$.*

Explicit mean-value filters. In general, for IVPs, 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

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

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 2.3) on the matrix $\Phi(t)$. We can write $\Phi(t) = m(\Phi(t)) + s(\Phi(t))$, and

$$m(\Phi(t))\mathbf{X} = \Gamma(t) - s(\Phi(t))\mathbf{X}. \tag{13}$$

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 (13) 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

$$m(\Phi(t))\mathbf{X} = \Gamma(t) - s(\Phi(t))s(\mathbf{D}^-). \tag{14}$$

Equation (14) can be rewritten as

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

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:

$$A_k(t)X_k = K(t) - \sum_{i=0}^{k-1} A_i(t)X_i. \tag{15}$$

Multiplying both sides of (15) by $A_k(t)^{-1}$ (recall that $A_k(t)^{-1}$ denotes an enclosure of the inverse of $A_k(t)$) gives

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

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*

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

where

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

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

$$D_k = m(D_k^-) + A_k(t)^{-1}K(t) - \sum_{i=0}^{k-1} \left(A_k(t)^{-1}A_i(t)\right) (D_i^* - m(D_i^*)), \tag{16}$$

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

$$D_k^* = D_k \cap D_k^-.$$

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

4.3. 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.

Wrapping effect. The wrapping effect is the name given to the overestimation that arises from enclosing a set by a box. In the context of ODEs, the set of solutions at each integration step is overapproximated by a box. These overapproximations 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 [20, 24]). How does the wrapping effect occur in our context? Let us rewrite an explicit mean-value filter from (16) as

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

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

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

The set Z is called a *zonotope*[8] (i.e., a generalization of a parallelepiped). Figure 3(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 3(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.

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

$$\begin{aligned} X_k &= K_0(e_0) + A_{0,0}(e_0) X_0 + \cdots + A_{0,k-1}(e_0) X_{k-1}, \\ X_{k+1} &= K_1(e_1) + A_{1,0}(e_1) X_1 + \cdots + A_{1,k-1}(e_1) X_k. \end{aligned}$$

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.

[8]Note that Kühn uses zonotopes in another context, i.e., as compact enclosures of solutions [17, 18, 19].

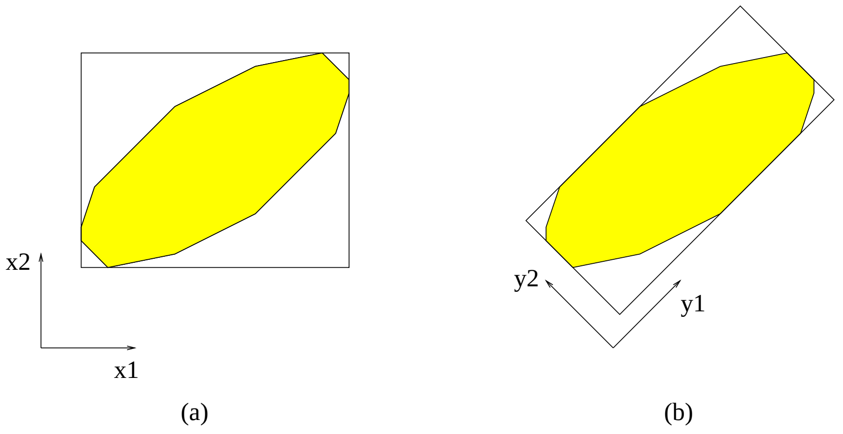

FIG. 3. (a) *A zonotope in* $\mathbb{R}^2$ *and the smallest enclosing box.* (b) *Coordinate transformation where the enclosing box better fits the zonotope.*

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.

4.4. 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 4 for $k = 3$. It can be seen that the global filter prunes the three 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^* are not 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* $\{FL_i(\mathbf{t}_{i..k+i}, \mathbf{D}_{i..k+i}, e_i)\}_{0\leq i<k}$ *of* k *successive explicit mean-value filters for* $\mathcal{O}$ *in* $\mathbf{D}^0_{0..k}, \ldots, \mathbf{D}^0_{k-1..2k-1}$ *respectively given as*

(17)
$$\begin{cases} X_k & = & K_0(e_0) + A_{0,0}(e_0)X_0 + \cdots + A_{0,k-1}(e_0)X_{k-1}, \\ X_{k+1} & = & K_1(e_1) + A_{1,0}(e_1)X_1 + \cdots + A_{1,k-1}(e_1)X_k, \\ & \vdots & \\ X_{2k-1} & = & K_{k-1}(e_{k-1}) + A_{k-1,0}(e_{k-1})X_{k-1} + \cdots + A_{k-1,k-1}(e_{k-1})X_{2k-2}, \end{cases}$$

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

The key idea to remove the variable dependency problem is to solve (17) globally by transforming the global filter into an explicit form

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

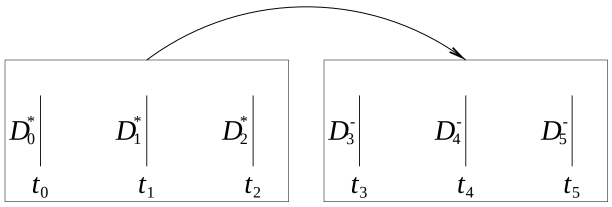

FIG. 4. *Intuition of the globalization process* ($k = 3$)*: Predicted boxes* D_3^-, D_4^-, *and* D_5^- *for times* t_3, t_4, *and* t_5 *are pruned globally using boxes* D_0^*, D_1^*, *and* D_2^* *computed for times* t_0, t_1, *and* t_2.

or, more concisely,

$$\mathbf{X}_1 = C(\mathbf{e}_0)\mathbf{X}_0 + R(\mathbf{e}_0), \tag{18}$$

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. Hence, it removes the dependencies introduced in D_3^- and D_4^-. Consider a system with $k = 3$:

$$\begin{cases} X_3 = A_{00}X_0 + A_{01}X_1 + A_{02}X_2 + K_0, \\ X_4 = A_{10}X_1 + A_{11}X_2 + A_{12}X_3 + K_1, \\ X_5 = A_{20}X_2 + A_{21}X_3 + A_{22}X_4 + K_2. \end{cases}$$

Variable X_4 can be eliminated from the last equation to obtain

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

To avoid multiplying interval matrices (e.g., $A_{22}A_{10}$), we can apply the midpoint technique (see point 1 of section 2.3) to obtain

$$\begin{aligned} X_5 = {} & A_{20}X_2 + A_{21}X_3 + m(A_{22})(A_{10}X_1 + A_{11}X_2 + A_{12}X_3 + K_1) \\ & + K_2 + s(A_{22})s(D_4^-). \end{aligned} \tag{19}$$

By distribution and rearrangement of the parentheses, we can rewrite (19) as

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

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 involving only X_4, X_0, X_1, and X_2.[9]

A generic algorithm for computing an explicit global filter is given in Figure 5. 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 w.r.t. the resulting boxes. Its precise specification is as follows.

[9]As observed by one of the reviewers, there are still some dependencies, but these are very small.

```
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
1     for $i := 0$ to $k-1$ do
2         $\langle K_i, A_{i,0}, \ldots, A_{i,k-1}\rangle := \text{EMVFL}(\mathbb{O}, \mathbf{t}_{i..i+k}, \mathbf{D}^0_{i..i+k}, \mathbf{B}_{i+1..i+k}, e_i)$;
3     endfor
4     for $i := k-1$ downto 0 do
5         $R_i := K_i$;
6         for $l := i$ downto 1 do
7             $A^* := m(A_{i,k-1})$;
8             $R_i := R_i + A^* K_{l-1} + s(A_{i,k-1}) s(D^0_{k+l-1})$;
9             for $j := k-1$ downto 1 do
10                $A_{i,j} := A_{i,j-1} + A^* A_{l-1,j}$
11            endfor
12            $A_{i,0} := A^* A_{l-1,0}$
13        endfor
14    endfor
15    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
```

FIG. 5. *An algorithm for computing an explicit global filter.*

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

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

$\mathbf{X}_0 = \mathbf{D}_0 - m(\mathbf{D}_0^0)$, *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 section 5, but its specification is given as follows.

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

$$\langle K(t), A_0(t), \ldots, A_{k-1}(t)\rangle = \text{EMVFL}(\mathbb{O}, \mathbf{t}, \mathbf{D}^0, (B_1, \ldots, B_k), t).$$

Then the interval relation

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

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, but 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.

4.5. 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 [20]. 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

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

or, equivalently, the system

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

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

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

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 accumulate drastically in subsequent integration steps. For this purpose, we introduce a coordinate transformation

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

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

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

We then solve the system

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

by inverting the matrix M_1:

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

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

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

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

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

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

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

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

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

to obtain the system

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

This equation system can be solved by inverting M_2:

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

Once again, observe the associativity in the multiplication to tackle the wrapping effect. The hope is that the matrix $M_2^{-1}(C_2(\mathbf{e}_1)M_1)$ is 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

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

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

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

The local coordinate transformation

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

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

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

In addition, in order to avoid the costly (see [15]) 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 2.3) to obtain

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

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

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

by the 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 [20] is a very successful scheme to obtain such a matrix.

4.6. 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 6, 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* $[t_{i-1}, t_i]$ *w.r.t.* (t_0, D_0) *for* $1 \leq i \leq 2k-1$. *Let*

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

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

```
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
```

FIG. 6. *The pruning algorithm on global filters.*

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 preceding section. It computes the explicit form of the global filter (line 1), the new boxes $\mathbf{X}_1$ (line 3), and the pruned boxes $\mathbf{D}_1^*$ (line 4). It then computes the new matrix M_1 (line 5) and the new boxes $\mathbf{Y}_1$ (line 7).

5. Hermite filters. In the previous section, 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. 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 [29, 1].

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

$$q^{(j)}(t_i) = j!(u_i)_j \qquad (0 \leq j \leq \sigma_i - 1,\ 0 \leq i \leq k). \tag{20}$$

PROPOSITION 5 (Hermite(σ) interpolation polynomial). *The polynomial* q *satisfying the conditions* (20) *is given by*

$$q(t) = \sum_{i=0}^{k} \sum_{j=0}^{\sigma_i - 1} j!(u_i)_j \varphi_{ij}(t), \tag{21}$$

where

$$\begin{aligned} &\varphi_{i,\sigma_i-1}(t) = l_{i,\sigma_i-1}(t), \quad i = 0, \dots, k, \\ &\varphi_{ij}(t) = l_{ij}(t) - \sum_{\nu=j+1}^{\sigma_i-1} l_{ij}^{(\nu)}(t_i)\varphi_{i\nu}(t), \quad i = 0, \dots, k, \ j = 0, \dots, \sigma_i - 2, \\ &l_{ij}(t) = \frac{(t-t_i)^j}{j!} \prod_{\substack{\nu=0 \\ \nu \neq i}}^{k} \left(\frac{t - t_\nu}{t_i - t_\nu}\right)^{\sigma_\nu}, \quad i = 0, \dots, k, \ j = 0, \dots, \sigma_i - 1. \end{aligned} \tag{22}$$

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, e.g., [21, 22, 27]). The only remaining issue is to bound the error terms. The following standard theorem (e.g., [29, 1]) provides the necessary theoretical basis.

THEOREM 1 (Hermite error term). *Let $p(\mathbf{t}, \mathbf{u}, t)$ be the Hermite(σ) interpolation polynomial in t w.r.t. f and $(\mathbf{t}, \mathbf{u})$. Let $u(t) = ms(\mathbf{t}, \mathbf{u}, t)$, $ms(\mathbf{t}, \mathbf{u}, t) = p(\mathbf{t}, \mathbf{u}, t) + e(\mathbf{t}, \mathbf{u}, t)$, $T = \square\{t_0, \dots, t_k, t\}$, $\sigma_s = \sum_{i=0}^{k} \sigma_i$, and $w(t) = \prod_{i=0}^{k}(t - t_i)^{\sigma_i}$. 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 do we use this theorem to bound the error terms? If B is a bounding box (produced by the bounding box process) for the ODE over $T = \square\{t_0, \dots, t_k, t\}$ w.r.t. $(\mathbf{t}_0, \mathbf{u}_0)$, it suffices to compute two boxes $(B)_{\sigma_s}$ and $(B)_{\sigma_s+1}$ by automatic differentiation. We then obtain

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

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). Reference [12] discusses how to evaluate Hermite polynomials accurately.

6. Optimal Hermite filters.

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

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

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

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

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

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

The purpose of this section 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.

6.1. 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 of 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 $u_0, \dots, u_{k-1}$ are given (and, of course, that $ms(\mathbf{t}_0, \mathbf{u}_0, t_k)$ is defined).[10]

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

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

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 6.2). To find an optimal evaluation time, we first derive the local error (section 6.3). From the local error, we can then characterize the optimal evaluation time (section 6.4). Two of the main results of this section 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 $(t_{i+1} - t_i)/h$ $(i = 0, \dots, k-1)$ between the interpolation points $t_0, \dots, t_k$ and on σ.[11] 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($(\sigma_0, \dots, \sigma_k)$) 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.

[10] As observed by one of the reviewers, the local error may be called more appropriately excess-width, since the enclosure contains the exact solution. We kept the term "local error" because of the analogy with traditional methods.

[11] Note that $h = t_k - t_0$ (and not $h = t_k - t_{k-1}$) because of the globalization process.

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

$$\exists c \in \mathbb{R} \quad \forall u, v \in \Omega : \|f(u) - f(v)\| \leq c\|u - v\|. \tag{23}$$

Note that (23) holds if we assume $f \in C^1(\Omega)$. We further assume that the interval extension F of function f satisfies ($D \subseteq \Omega$)

$$\omega(F(D)) = \mathcal{O}(\omega(D)). \tag{24}$$

For instance, (24) holds if F is the natural interval extension of f and (23) holds. We also assume that B is a bounding box of $u' = f(u)$ over $T = \Box\{t_0, \ldots, t_k, t\}$ w.r.t. $(\mathbf{t}_0, \mathbf{u}_0)$ and that (see [23])

$$\omega\left((B)_j\right) = \Theta(\omega(B)) = \Theta(h), \qquad j \in \mathbb{N}. \tag{25}$$

From (23), the condition (25) holds if $(B)_j$ is a sufficiently tight enclosure of the set $\{(x)_j \mid x \in B\}$. 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

$$FL(\mathbf{t}, (\mathbf{u}_0, v), t) \Rightarrow \frac{\partial p}{\partial t}(\mathbf{t}, (\mathbf{u}_0, v), t) + DE(t) - F(p(\mathbf{t}, (\mathbf{u}_0, v), t) + E(t)) = 0, \tag{26}$$

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 w.r.t. f and $(\mathbf{t}, (\mathbf{u}_0, v))$.

Let us introduce the function

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

where $m_e(t) = m(E(t))$. From the hypothesis (24), the condition (26) can be rewritten as

$$FL(\mathbf{t}, (\mathbf{u}_0, v), t) \Rightarrow \delta(\mathbf{t}, (\mathbf{u}_0, v), t) = -DE(t) + \mathcal{O}(\omega(E(t))). \tag{27}$$

In case (24), the condition (27) is satisfied for natural Hermite filters (see section 4.1), 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 paper.) If we assume that the interval extension of the Jacobian of f satisfies the same condition as F, i.e., $\omega(\mathcal{J}(D)_0) = \mathcal{O}(\omega(D))$, then (27) is satisfied for *implicit mean-value* Hermite filters. It is also a good approximation for *explicit mean-value* Hermite filters if the matrix inversion is accurate (see section 4.2). We will also denote the Jacobian of δ w.r.t. variable v by

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

Finally, we introduce the following functions:

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

6.3. Local error of a natural 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 [12]. We are now in position to characterize the local error of a Hermite filter.

THEOREM 2 (local error of a Hermite filter). *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying* (27). *We have*

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

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

$$FL(\mathbf{t},(\mathbf{u}_0,v_1),t) \ \text{ and } \ FL(\mathbf{t},(\mathbf{u}_0,v_2),t).$$

By the mean-value theorem, we can write

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

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 (27)

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

Since the two vectors v_1 and v_2 are chosen arbitrarily, it follows from (25) that

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

which proves point 1. Points 2 and 3 are now direct consequences of Lemma 1 and (25). ∎

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

6.4. Optimal evaluation time for a natural Hermite filter. Our first result 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 3 (order of a Hermite filter). *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying* (27)*. 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 2, $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \mathcal{O}(h^{\sigma_s+2})$. Now assume that $w'(t) \neq 0$. By Theorem 2, $e_{loc}(FL, \mathbf{t}_0, \mathbf{u}_0, t) = \Omega(h^{\sigma_s+1})$. ∎

Theorem 3 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 4 (necessary condition for optimal Hermite filters). *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying* (27)*, and let $t_e \in \mathbb{R}$ be such that*

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

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 3, $w'(t_e)$ 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 $w'(t_i) = 0$ $(0 \leq i \leq k)$. Thus $t_e \notin \{t_0, \dots, t_k\}$ 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 4 has exactly k zeros $s_0, \dots, s_{k-1}$ such that $t_i < s_i < t_{i+1}$ $(0 \leq i < k)$.*

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 $w(s_i) \neq 0$ $(0 \leq i < k)$. Furthermore, the roots of w' are in $\{s_0, \dots, s_{k-1}, t_0, \dots, t_k\}$ because t_i is a root of multiplicity $\sigma_i - 1$ $(0 \leq i \leq k)$ 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 $t_0, \dots, t_k$, its zeros are in $\{s_0, \dots, s_{k-1}\}$. ∎

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

Theorem 5 (optimal evaluation time). *Let FL be a Hermite(σ) filter for $u' = f(u)$ satisfying* (27)*, let $s_0 < \dots < s_{k-1}$ be the zeros of γ, and let $t_e \in \mathbb{R}$ such that*

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

Then, for h sufficiently small,

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

TABLE 1
Relative distance between the rightmost zero t_e of γ and t_k when $\sigma_0 = \cdots = \sigma_k$.

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

Proof. Let us assume that h is sufficiently small. From Theorem 4, 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 2,

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

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

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

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

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

which is a contradiction. $\Box$

6.5. Discussion. It is important to discuss the consequences of Theorem 4 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 $\sigma_0 = \cdots = \sigma_k$, 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 rightmost zero was always the optimal evaluation time when $\sigma_0 = \cdots = \sigma_k$, although we have not been able to prove this result.

6.6. Illustration. We now illustrate the theoretical results presented in this section. Table 1 gives approximative values of the relative distance $(t_e - t_k)/h$ between the rightmost zero t_e of the function γ and the point t_k $(1 \leq k \leq 6)$ for $\sigma_0 = \cdots = \sigma_k$ and $t_{i+1} - t_i = h/k$ $(i = 0, \ldots, k-1)$. 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.

Figure 7 illustrates the functions γ, w, w', λ, and w/λ for $k = 4$ and $\sigma = (2, 2, 2, 2, 2)$. The top left figure shows the function w' and γ, as well as the zeros of

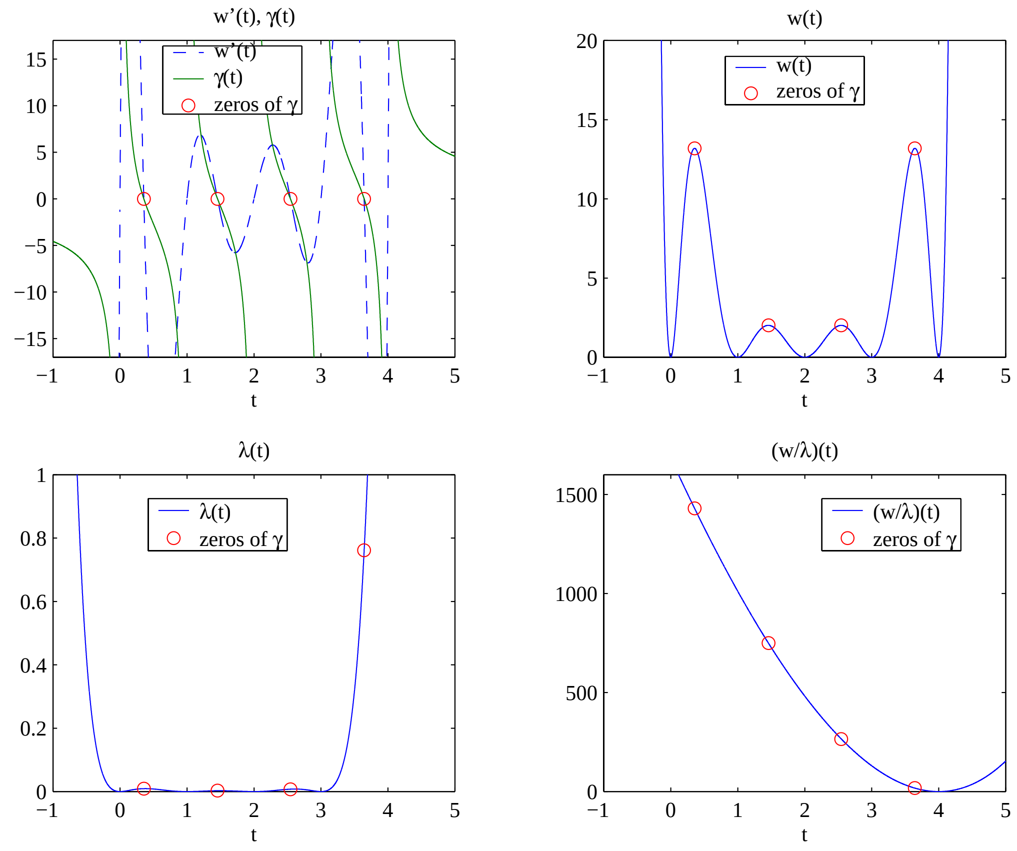

FIG. 7. *The functions* γ, w, w', λ *and* w/λ *for the case* $k = 4, \sigma = (2, 2, 2, 2, 2)$.

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 rightmost zero minimizes the local error in this example.

6.7. 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 limits the stability of validated methods and makes stiff problems more challenging;
2. the existing bounding box process, which often impose the strongest restriction on the step size, especially for stiff problems.

Figure 8 illustrates our theoretical results experimentally on a specific ODE. It plots the local error of several global Hermite filters (GHFs) as a function of the evaluation time for the Lorenz system (e.g., [10]). 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 GHF, the distance between the evaluation time and the rightmost interpolation point

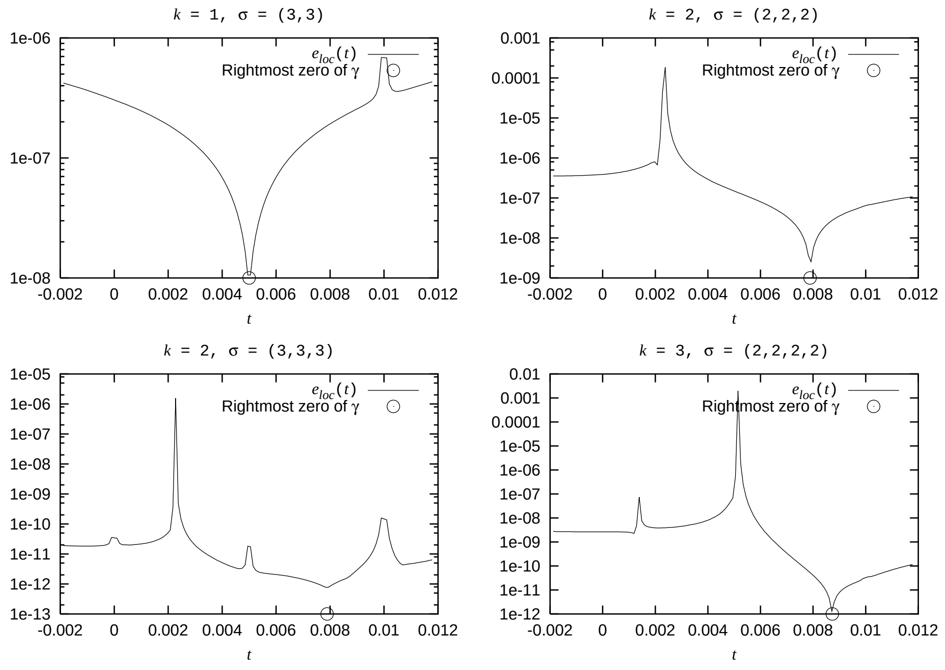

FIG. 8. *Local error of GHFs as a function of the evaluation time for the Lorentz system.*

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 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.

7. The algorithm. We are now in position to present our algorithm for enclosing solutions of IVPs for parametric ODEs. The algorithm is presented in Figure 9, and Figure 10 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 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.

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

8.1. Overview of the methods. We analyze the cost of our SOLVE algorithm based on the GHF method and compare it to Nedialkov's interval Hermite–Obreschkoff (IHO) method [24], the best interval method we know of. Indeed, the IHO method outperforms interval Taylor series methods such as Lohner's method [20]. Here are the various methods used in the theoretical and experimental comparisons.

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 $\sigma_s = \sigma_0 + \cdots + \sigma_k$, the remaining components of GHF are specified as follows:

```
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
```

FIG. 9. *The constraint satisfaction algorithm for IVPs 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 $1 \le i \le mk-1$, $s(t_0, D_0, t_i) \subseteq D_i$.*
SPECIFICATION 5 (BOUNDINGBOX). *Let $\mathbf{B}_{1..k}$ =BOUNDINGBOX$(\mathbb{O}, t_0, D_0, \mathbf{t}_{1..k})$. Then, for $1 \le i \le k$, B_i is a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ w.r.t. (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 $[t_{i-1}, t_i]$ w.r.t. (t_0, D_0) for $1 \le i \le k-1$. Let*

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

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 $0 \le i \le k-1$, $ms(t_0, D_0, t_i) \subseteq ms(\mathbf{t}_0, \mathcal{A}, t_i)$.
SPECIFICATION 7 (PREDICTOR). *Let s be the solution of ODE $\mathbb{O}$ and B_i a bounding box of $\mathbb{O}$ over $[t_{i-1}, t_i]$ w.r.t. (t_0, D_0) for $1 \le i \le k$. Let*

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

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

FIG. 10. *The specification of the main functions.*

1. The BOUNDINGBOX function in GHF uses a Taylor series method [21, 5, 25] of order $p+q+1$ to compute $\mathbf{B}_i$. Moreover, we assume that $B_{ik} = \cdots = B_{(i+1)k-1}$; i.e., the function computes a single bounding box over $[t_{ik-1}, t_{(i+1)k-1}]$ $(i \ge 1)$.
2. The PREDICTOR function uses Moore's Taylor method [21] 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 section 6 and Table 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 $(i-1)k \le j \le (i+1)k - 1$ to compute the k explicit mean-value Hermite filters in EMVFL. GHF computes only Jacobians at predicted boxes and not at pruned boxes. More precisely, it computes only $k(\sigma_m - 1)$ 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 [20]).
6. The function InitializeMultistep uses a one-step mean-value Taylor method.

The IHO method. The IHO method is implemented exactly as described in [24]. 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.

The IHO* method. To obtain experimental results as informative as possible, we introduce IHO*, a variant of IHO that 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. IHO* and GHF differ only 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.

8.2. 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.[12] 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 [15]). 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.

8.3. 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 2 reports the main cost of a step in IHO, IHO*, and GHF. 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 ($\sigma_0 = \cdots = \sigma_k = 2$).

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 section shows that an improvement in accuracy is also obtained in this case.

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

TABLE 2
Cost analysis: Methods of the same order.

	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 3
Cost analysis: Methods of different orders but of similar cost.

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

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 w.r.t. 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 tailored to the application at hand.*

8.4. 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 method of order $\sigma_s + 1$. Table 3 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.

9. Experimental analysis. We now report experimental results of a C++ implementation[13] of our SOLVE algorithm based on the GHF 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 [15] and FADBAD/TADIFF [3, 2].

The benchmarks. Many of the benchmarks are standard. They come from various domains, including chemistry, biology, mechanics, physics, and electricity. The equation, initial conditions, and interval of integration for each IVP are given in [12]. Note that the comparisons uses only 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) exemplifies the so-called strange at-

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

tractors. 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 [10, 11]. We also consider a problem from molecular biology (BIO) and the Stiff DETEST problem D1 [9]. Finally, we consider four dynamical systems (LIEN, P1, P2, P3), where the function f contains more operations. LIEN, P2, and P3 are taken from [26].

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

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.

9.1. One-step methods.

Same order. Table 4 reports the experimental results for the IHO$^{(*)}(p,p)$ and 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 w.r.t. the step size, confirming that GHF and IHO$^{(*)}$ are methods of the same order.

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 5 compares IHO(p,p) (order $2p+1$) and 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.

Error w.r.t. time. It is interesting to compare the various methods by plotting the error as a function of the execution time. Figure 11 plots IHO$^{(*)}(p,p)$, GHF(p,p), and GHF$(p+1,p+1)$ using the results in Tables 4 and 5. 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.

Table 4
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	(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 5
One-step methods of different orders.

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

9.2. 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 our attention to problems where the function f contains more operations. Tables 6, 7, 8, and 9 report the results, respectively, for the four tested examples and for several orders and step sizes.[14] 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

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

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 12, 13, 14, and 15 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.

9.3. Discussion. Before concluding this section, it is important to make a number of remarks.

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.

As pointed out by Nedialkov [23], the stability of interval methods depends not only on the stability of the underlying approximation formula (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.

In our experiments, we always chose $\sigma_0 = \cdots = \sigma_k$. 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 $\sigma_0 = \cdots = \sigma_k$. 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.

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, and P2).

9.4. Summary. We now summarize our experimental results. The main conclusions are as follows:

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

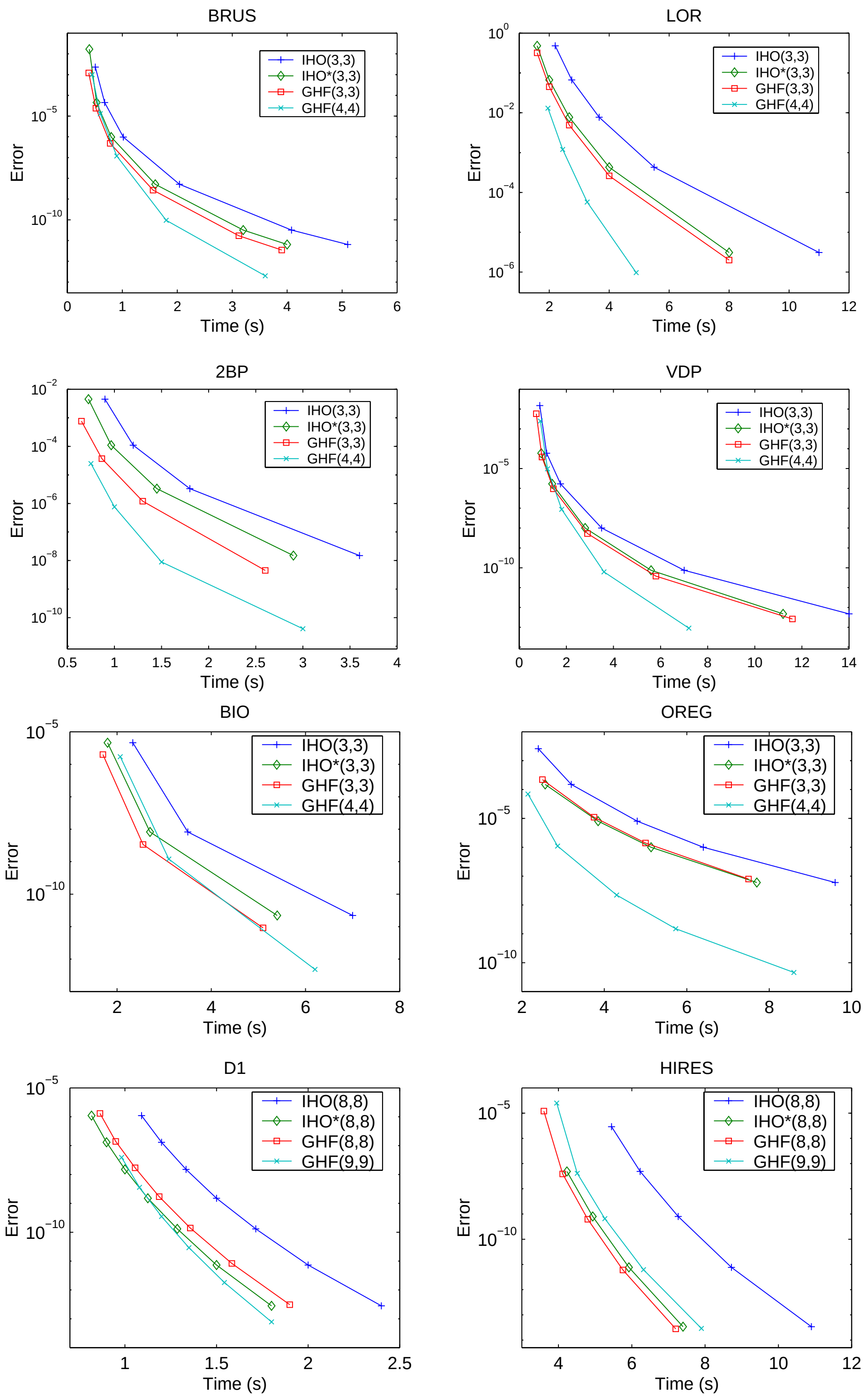

FIG. 11. *Comparison of the methods* $IHO^{(*)}(p,p)$, $GHF(p,p)$ *and* $GHF(p+1,p+1)$ *for the problems BRUS, LOR, 2BP, VDP, BIO, OREG, D1, and HIRES.*

Table 6
Multistep versus one-step methods: The LIEN problem.

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

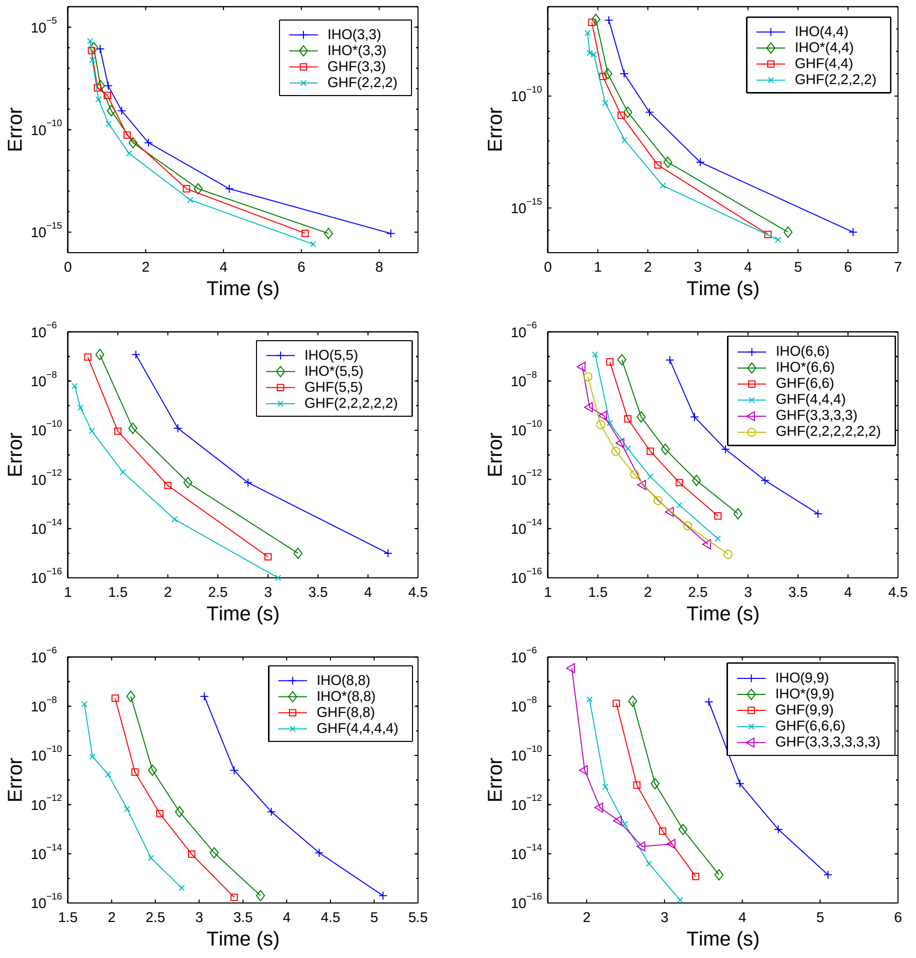

FIG. 12. *Multistep versus one-step methods: The LIEN problem.*

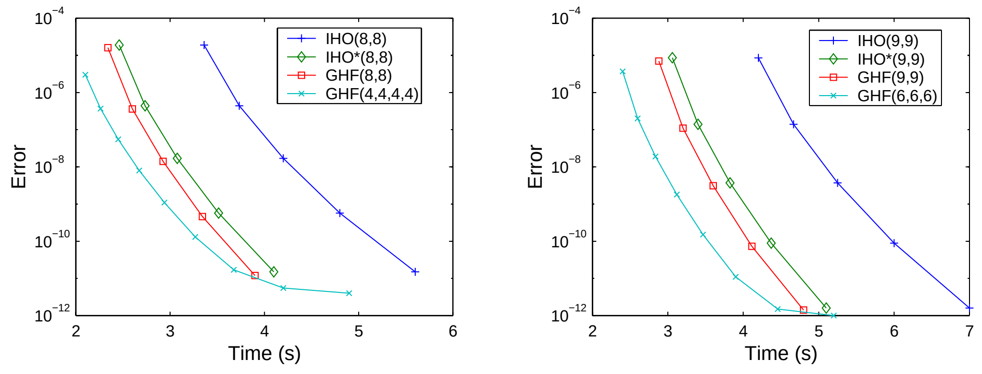

FIG. 13. *Multistep versus one-step methods: The $P1$ problem.*

Table 7
Multistep versus one-step methods: The $P1$ 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

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 [8] or Nedialkov's [23] techniques.

10. Conclusion.

This paper described a constraint satisfaction approach to IVPs for parametric ODEs (i.e., ODEs where some data or initial conditions are

Table 8
Multistep versus one-step methods: The $P2$ 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 9
Multistep versus one-step methods: The $P3$ 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

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).* Then we presented an effective pruning algorithm which uses (1) relaxations of the ODE based Hermite interpolation polynomials and enclosures of their error terms; (2) a globalization process to reduce variable dependency problems and evaluation points that minimize the local error of the relaxations. The pruning component

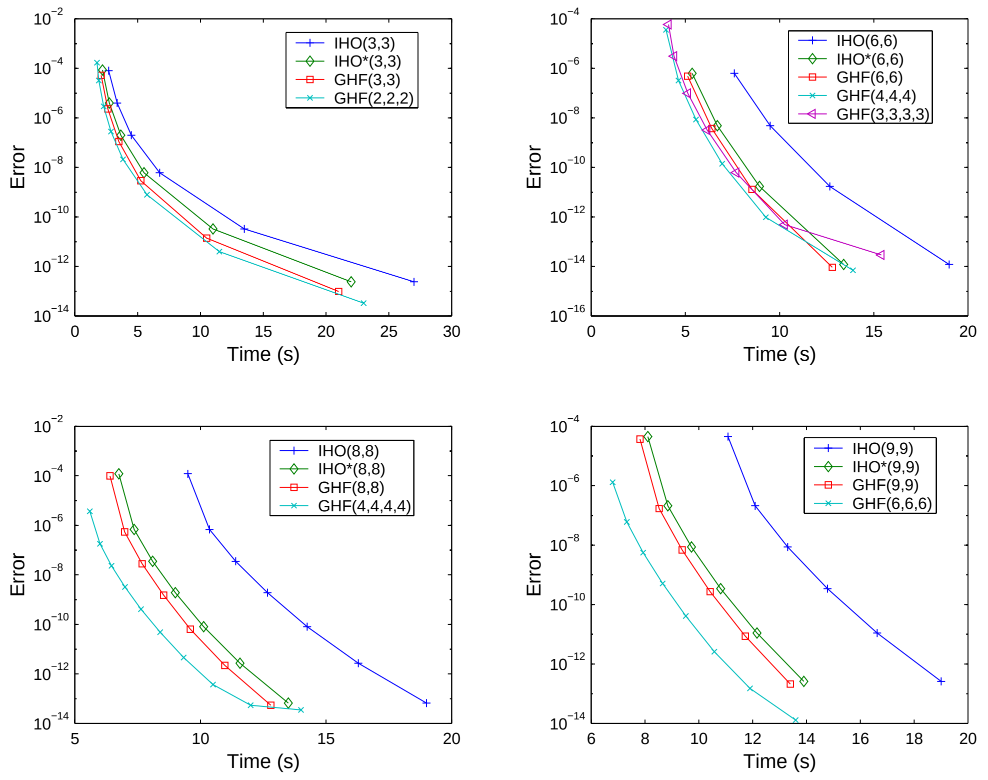

FIG. 14. *Multistep versus one-step methods: The $P2$ problem.*

was integrated in an integration algorithm which also uses traditional techniques to handle the wrapping effect.

The novel 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 accuracy 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, GHFs 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 enhanced in many ways to include, for instance, order and step size control strategies, and the 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 relax-

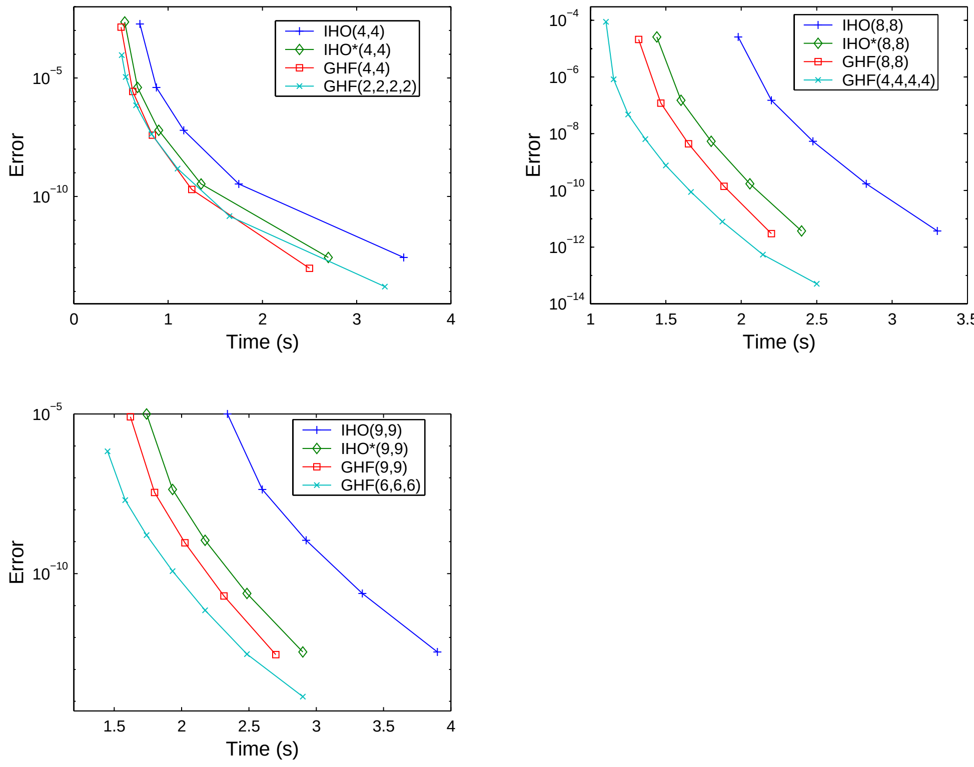

FIG. 15. *Multistep versus one-step methods: The P3 problem.*

ations (e.g., using splines, trigonometric interpolation, Legendre, Chebyshev, and Laguerre polynomials) should be investigated.

3. Compared to standard numerical methods, validated methods generally use smaller step sizes, and stiff problems are particularly challenging. The main factors that limit the step size are the need to enclose error terms and the bounding box process. Finding efficient bounding box techniques is probably the main bottleneck at this point, and it would be interesting to study how pruning techniques could help in this respect. Once we will be able to increase the step size, it will be important to analyze the stability of our approach and to compare it to the stability of other validated methods. The choice of many of the parameters mentioned in point (1) will be guided by stability requirements in the case of stiff problems. Furthermore, our asymptotic theory for choosing an optimal evaluation time may not be valid anymore, and we may 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 can thus keep the parametric aspect of the ODEs, but we lose 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 $\sigma_i \leq 2,\ i = 0, \dots, k$, is the *only* interval method (we know of) 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 $\sigma = (1, \dots, 1)$, 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.

In summary, the constraint satisfaction approach should be a valuable addition to existing methods for the reliable solutions of differential equations, and there is considerable room for further research in this area.

Acknowledgments. We give special thanks to Philippe Delsarte for interesting discussions and for his detailed comments. We also would like to express our gratitude to the reviewers for their detailed suggestions.

REFERENCES

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

[2] 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, Kgs. Lyngby, Denmark, 1997.

[3] 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, Kgs. Lyngby, Denmark, 1996.

[4] M. Berz and K. Makino, *Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models*, Reliab. Comput., 4 (1998), pp. 361–369.

[5] G. F. Corliss and R. Rihm, *Validating an a priori enclosure using high-order Taylor series*, in Scientific Computing, Computer Arithmetic, and Validated Numerics, Akademie Verlag, Berlin, 1996, pp. 228–238.

[6] J. Cruz and P. Barahona, *An interval constraint approach to handle parametric ordinary differential equations for decision support*, in Proceedings of EKBD-99, 1999, pp. 93–108.

[7] Y. Deville, M. Janssen, and P. Van Hentenryck, *Consistency techniques in ordinary differential equations*, in Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming, Pisa, Italy, 1998.

[8] P. Eijgenraam, *The Solution of Initial Value Problems Using Interval Arithmetic*, Math. Centre Tracts 144, Stichting Mathematisch Centrum, Amsterdam, The Netherlands, 1981.

[9] W. H. Enright, T. E. Hull, and B. Lindberg, *Comparing numerical methods for stiff systems of ODEs*, BIT, 15 (1975), pp. 10–48.

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

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

[12] M. Janssen, *A Constraint Satisfaction Approach for Enclosing Solutions to Parametric Ordinary Differential Equations*, Ph.D. thesis, Department of Computer Science, UCL, Louvain, Belgium, 2001; also available online at http://www.info.ucl.ac.be.

[13] M. Janssen, Y. Deville, and P. Van Hentenryck, *Multistep filtering operators for ordinary differential equations*, in Proceedings of the Fifth International Conference on Principles and Practice of Constraint Programming, Alexandria, VA, 1999.

[14] M. Janssen, P. Van Hentenryck, and Y. Deville, *A constraint satisfaction approach to parametric differential equations*, in Proceedings of the Joint International Conference on Artificial Intelligence, Seattle, WA, 2001.

[15] O. Knüppel, *PROFIL/BIAS — a fast interval library*, Computing, 53 (1994), pp. 277–287.

[16] F. Krueckeberg, *Ordinary differential equations*, in Topics in Interval Analysis, E. Hansen, ed., Clarendon Press, Oxford, UK 1969, pp. 91–97.

[17] W. Kühn, *Rigorously computed orbits of dynamical systems without the wrapping effect*, Computing, 61 (1998), pp. 47–67.

[18] W. Kühn, *Zonotope dynamics in numerical quality control*, in Mathematical Visualization. Algorithms, Applications, and Numerics, H.-Ch. Hege and K. Polthier, eds., Springer-Verlag, Berlin, 1998, pp. 125–134.

[19] W. Kühn, *Towards an optimal control of the wrapping effect*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, Dordrecht, The Netherlands 1999, pp. 43–51.

[20] R. J. Lohner, *Enclosing the solutions of ordinary initial and boundary value problems*, in Computer Arithmetic: Scientific Computation and Programming Languages, Teubner, Stuttgart, 1987.

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

[22] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM Stud. Appl. Math. 2, SIAM, Philadelphia, 1979.

[23] N. S. Nedialkov, *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*, Ph.D. thesis, Computer Science Department, University of Toronto, Toronto, ON, Canada, 1999.

[24] 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*, in Developments in Reliable Computing, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 289–310.

[25] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, *Validated solutions of initial value problems for ordinary differential equations*, Appl. Math. Comput., 105 (1999), pp. 21–68.

[26] L. Perko, *Differential Equations and Dynamical Systems*, Springer-Verlag, New York, 2000.

[27] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Comput. Sci. 120, Springer-Verlag, Berlin, 1981.

[28] R. Rihm, *Implicit methods for enclosing solutions of ODEs*, J. UCS, 4 (1998).

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

[30] P. Van Hentenryck, D. McAllester, and D. Kapur, *Solving polynomial systems using a branch and prune approach*, SIAM J. Numer. Anal., 34 (1997), pp. 797–827.

[31] P. Van Hentenryck, L. Michel, and Y. Deville, *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA, 1997.

[32] P. Van Hentenryck, *A gentle introduction to Numerica*, Artificial Intelligence, 103 (1998), pp. 209–235.