Validated Solutions of Initial Value Problems for Parametric ODEs

Youdong Lin and Mark A. Stadtherr*
Department of Chemical and Biomolecular Engineering
University of Notre Dame, Notre Dame, IN 46556, USA

(revised, October 2006)
Accepted for publication in *Applied Numerical Mathematics*

Abstract

In initial value problems for ODEs with interval-valued parameters and/or initial values, it is desirable in many applications to be able to determine a validated enclosure of all possible solutions to the ODE system. Much work has been done for the case in which initial values are given by intervals, and there are available software packages that deal with this case. However, less work has been done on the case in which parameters are given by intervals. We describe here a new method for obtaining validated solutions of initial value problems for ODEs with interval-valued parameters. The method also accounts for interval-valued initial values. The effectiveness of the method is demonstrated using several numerical examples involving parametric uncertainties.

Keywords: Ordinary differential equations; Interval analysis; Validated computing; Taylor models; Dynamic systems

1 Introduction

Initial value problems (IVPs) for ordinary differential equations (ODEs) arise naturally in many applications in science and engineering. It is often the case that the problem involves parameters and/or initial values that are not known with certainty but that can be expressed as intervals. For this situation it is desirable to be able to determine an enclosure of all possible solutions to the ODEs. Traditional approximate solution methods for ODEs are not useful in this context, since, in essence, they would have to solve infinitely many systems to determine such an enclosure. Interval methods [25] for ODEs, on the other hand, provide a natural approach for computing the desired enclosure. Even in the case in which the initial values and parameters are known exactly, standard numerical methods for solving ODEs only compute an approximate solution to some tolerance, without guaranteed error bounds, and so it is of interest to determine an enclosure of the true solution. Interval methods (also called validated methods or verified methods) not only can determine a guaranteed error bound on the true solution, but can also verify that a unique solution to the problem exists. An excellent review of interval methods for IVPs has been given by Nedialkov

*Author to whom all correspondence should be addressed. Phone: (574) 631-9318; Fax: (574) 631-8366; E-mail: markst@nd.edu

et al. [28]. Much work has been done for the case in which the initial values are given by intervals, and there are several available software packages that deal with this case. However, relatively little work has apparently been done on the case in which parameters are given by intervals. We concentrate here on the case of such parametric ODEs. However, the method developed will also account for interval-valued initial values.

Traditional interval methods usually consist of two processes applied at each integration step [28]. In the first process, existence and uniqueness of the solution are proven, and a rough enclosure of the solution is computed. In the second process, a tighter enclosure of the solution is computed. This can be regarded as similar to the familiar predictor-corrector procedure employed in many standard methods for obtaining an approximate solution. In general, both processes are realized by applying interval Taylor series (ITS) expansions with respect to time, and using automatic differentiation to obtain the Taylor coefficients. A major difficulty in interval methods is the overestimation of bounds caused by the dependency problem of interval arithmetic and by the wrapping effect [25]. The accumulation of overestimations at successive time steps may ultimately lead to an explosion of enclosure sizes, causing the integration procedure to abort. Several schemes for reducing the overestimation of bounds have been proposed. For example, Lohner's AWA package employs a QR-factorization method which features efficient coordinate transformations to tackle the wrapping effect [15]. Nedialkov's VNODE package employs QR together with an interval Hermite-Obreschkoff method [27, 29], which can be viewed as a type of generalized Taylor method, and improves on AWA. Janssen et al. [7] have introduced a constraint satisfaction approach to these problems, which enhances traditional interval methods with a pruning step based on a global relaxation of the ODEs. Another approach for addressing the dependency problem and the wrapping effect has been described by Berz and Makino [3] and implemented in the beam dynamics package COSY INFINITY. This scheme is based on expressing the dependence on initial values and time using a Taylor model (i.e., a Taylor polynomial and an interval remainder bound). Neher et al. [30] have recently described this Taylor model approach in some detail and compared it to traditional interval methods. We will also use Taylor models in the new method described here, though they will be determined and used in a different way, and a new type of Taylor model will be introduced.

Available general-purpose validated ODE solvers are focused on dealing with uncertainties in the initial values. Some solvers, including VNODE [26], can take interval parameters as input. These solvers, however, can become very inefficient in the presence of interval parameters because this tends to exacerbate the wrapping effect; thus the size of the enclosure can grow so quickly that the integration must be stopped. An alternative approach is to treat time-invariant interval parameters as additional state variables, with zero first-order derivatives, as suggested by Lohner [16]. Since the parameters are now treated as independent variables, tighter enclosures can be obtained using this approach. However, the increase in the number of state variables can result in a significant increase in the computational expense. For example, in a problem with n states and p parameters, an $(n+p) \times (n+p)$ matrix, instead of an $n \times n$ matrix, must be factored at each time step in the usual approach for controlling the wrapping effect. In the work described here, we will develop a method for efficiently determining validated solutions of ODEs with parametric uncertainties, where instead of increasing the number of state variables, we will treat the parametric uncertainties directly. The method follows the traditional two-phase interval approach, but makes use, in a novel way, of Taylor models.

Singer and Barton [35] have recently suggested another approach for bounding the solutions of parametric ODEs. They use convex underestimators and concave overestimators to construct two

bounding IVPs, which are then solved to obtain the lower and upper bounds on the trajectories. However, as implemented [34], the bounding IVPs are solved using standard numerical methods that do not provide guaranteed error estimates. Thus, this approach cannot be regarded as providing rigorously guaranteed enclosures.

The remainder of this paper is organized as follows. Section 2 describes the problem we are addressing and the notation used. Section 3 provides background on interval analysis and a brief introduction to the use of Taylor models, based on the Taylor model arithmetic (remainder differential algebra) described by Makino and Berz [20, 23]. Section 4 presents the new interval method for obtaining guaranteed enclosures of the solutions of parametric ODEs. Then, in Section 5 we present the results of several numerical experiments that demonstrate the effectiveness of the new method.

2 Problem Description

In this section we describe the problem to be solved, and introduce our notation. We consider validated numerical methods for solving the set of parametric autonomous IVPs

$$y'(t) = f(y, \theta), \quad y(t_0) = y_0 \in Y_0, \quad \theta \in \Theta, \tag{1}$$

where $t \in [t_0, t_m]$ for some $t_m > t_0$. Here θ is a p-dimensional vector of time-invariant parameters, y is the n-dimensional vector of state variables, and y_0 is the n-dimensional vector of initial values. The interval vectors Θ and Y_0 represent enclosures of the uncertainties in θ and y_0, respectively. Uppercase will be used to denote interval-valued quantities, unless noted otherwise. We assume that $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is $(k-1)$-times continuously differentiable with respect to the state variables y on $\mathbb{R}^n$, and $(q+1)$-times continuously differentiable with respect to the parameters θ on $\mathbb{R}^p$. Here, k is the order of the truncation error in the ITS method to be used, and q is the order of the Taylor model to be used to represent dependence on the uncertain quantities (parameters and/or initial values). We also assume that the representation of f contains a finite number of constants, variables, elementary operations, and standard functions. Because of the differentiability requirement, functions such as branches, abs, min or max are excluded.

A sequence of values, $t_0 < t_1 < \cdots < t_m$ will be considered, with step size $h_j = t_{j+1} - t_j$ (not necessarily constant) at the $(j+1)$-th integration step, $j = 0, \ldots, m-1$. A solution of $y'(t) = f(y, \theta)$ for the initial condition $y = y_j$ at $t = t_j$ is denoted by $y(t; t_j, y_j, \theta)$. We denote by $y(t; t_j, Y_j, \Theta)$ the set of solutions

$$y(t; t_j, Y_j, \Theta) = \{y(t; t_j, y_j, \theta) \mid y_j \in Y_j, \theta \in \Theta\}. \tag{2}$$

In this work, we seek to determine enclosures Y_j, $j = 1, \ldots, m$, of the state variables at each time step such that

$$y(t_j; t_0, Y_0, \Theta) \subseteq Y_j, \tag{3}$$

and to do so in such a way that there will be relatively little overestimation.

Traditional interval methods for solving (1) involve the use of Taylor series expansions with respect to time. In a Taylor expansion of $y(t)$ with respect to time, the i-th Taylor coefficient evaluated at t_j is denoted by

$$(y_j)_i = \frac{y^{(i)}(t_j)}{i!}, \tag{4}$$

where $y^{(i)}(t)$ is the i-th derivative of $y(t)$. In terms of $f(y,\theta) = y'(t)$, these Taylor coefficients can be expressed recursively using functions denoted by $f^{[i]}$ as

$$(y_j)_0 = f^{[0]}(y_j,\theta) = y_j \tag{5}$$

$$(y_j)_1 = f^{[1]}(y_j,\theta) = f(y_j,\theta) \tag{6}$$

$$(y_j)_i = f^{[i]}(y_j,\theta) = \frac{1}{i}\left(\frac{\partial f^{[i-1]}}{\partial y} f\right)(y_j,\theta), \qquad i \geq 2. \tag{7}$$

This allows for the efficient generation of Taylor coefficients [25]. Interval extensions, $(Y_j)_i = F^{[i]}(Y_j,\Theta)$, of the Taylor coefficients for $y_j \in Y_j$ and $\theta \in \Theta$ can be generated by using the same procedure in interval arithmetic, as discussed in the next section, or by using other methods, as described in Section 4.

3 Background

3.1 Interval analysis

A real interval C is defined as the set of real numbers lying between (and including) given upper and lower bounds; that is,

$$C = \left[\underline{C}, \overline{C}\right] = \left\{c \in \mathbb{R} \mid \underline{C} \leq c \leq \overline{C}\right\}. \tag{8}$$

Here an underline is used to indicate the lower bound of an interval and an overline is used to indicate the upper bound. Thus, uppercase quantities with underline or overline are real. The set of all real intervals is denoted by $\mathbb{IR}$. For an interval $C = [\underline{C}, \overline{C}]$, the *width* is denoted by $w(C) = \overline{C} - \underline{C}$ and the *midpoint* by $m(C) = (\overline{C} + \underline{C})/2$. A real interval vector $X = (X_1, X_2, \cdots, X_n)^T \in \mathbb{IR}^n$ has n real interval components and can be interpreted geometrically as an n-dimensional rectangle or box. Interval matrices can be similarly defined. For an interval vector or matrix, the width and midpoint are defined componentwise.

Basic arithmetic operations with intervals are defined by

$$X \text{ op } Y = \{x \text{ op } y \mid x \in X, y \in Y\}, \tag{9}$$

where op $= \{+, -, \times, \div\}$. Interval versions of the elementary functions can be similarly defined. It should be emphasized that, when machine computations with interval arithmetic operations are done, as in the procedures outlined below, the endpoints of an interval are computed with a directed (outward) rounding. That is, the lower endpoint is rounded down to the next machine-representable number and the upper endpoint is rounded up to the next machine-representable number. In this way, through the use of interval, as opposed to floating-point arithmetic, any potential rounding error problems are avoided. Several good introductions to interval analysis, as well as interval arithmetic and other aspects of computing with intervals, are available [8, 6, 9, 31]. Implementations of interval arithmetic and elementary functions are also readily available, and recent compilers from Sun Microsystems directly support interval arithmetic and an interval data type.

For an arbitrary function $g(x)$, the *interval extension*, denoted by $G(X)$, encloses all possible values of $g(x)$ for $x \in X$. That is, $G(X) \supseteq \{g(x) \mid x \in X\}$ encloses the range of $g(x)$ over X. It is

often computed by substituting the given interval X into the function $g(x)$ and then evaluating the function using interval arithmetic. This "natural" interval extension may be wider than the actual range of function values, although it always includes the actual range. For example, the natural interval extension of $g(x) = x/(x-1)$ over the interval $X = [2,3]$ is $G([2,3]) = [2,3]/([2,3]-1) = [2,3]/[1,2] = [1,3]$, while the true function range over this interval is $[1.5,2]$. This overestimation of the function range is due to the "dependency" problem of interval arithmetic, which may arise when a variable occurs more than once in a function expression. While a variable may take on any value within its interval, it must take on the *same* value each time it occurs in an expression. However, this type of dependency is not recognized when the natural interval extension is computed. In effect, when the natural interval extension is used, the range computed for the function is the range that would occur if each instance of a particular variable were allowed to take on a different value in its interval range. For the case in which $g(x)$ is a single-use expression, that is, an expression in which each variable occurs only once, natural interval arithmetic will always yield the true function range. For example, rearrangement of the function expression used above gives $g(x) = x/(x-1) = 1 + 1/(x-1)$, and now $G([2,3]) = 1 + 1/([2,3]-1) = 1 + 1/[1,2] = 1 + [0.5,1] = [1.5,2]$, the true range. There are a variety of approaches that can be used to try to avoid the overestimation that may occur due to the dependency problem, including the use of Taylor models, as described in the next subsection.

3.2 Taylor models

Makino and Berz [20] have described a remainder differential algebra (RDA) approach for bounding function ranges and control of the dependency problem of interval arithmetic [21]. In this method a function is represented using a model consisting of a Taylor polynomial and an interval remainder bound.

One way of forming a Taylor model of a function is by using a truncated Taylor series. Consider a function $f : x \in X \subset \mathbb{R}^m \to \mathbb{R}$ that is $(q+1)$ times partially differentiable on each component of x, and let $x_0 \in X$. The Taylor theorem states that for each $x \in X$, there exists a $\zeta \in \mathbb{R}$ with $0 < \zeta < 1$ such that

$$f(x) = \sum_{i=0}^{q} \frac{1}{i!} \left[(x - x_0) \cdot \bigtriangledown\right]^i f(x_0) + \frac{1}{(q+1)!} \left[(x - x_0) \cdot \bigtriangledown\right]^{q+1} f\left[x_0 + (x - x_0)\zeta\right], \tag{10}$$

where the partial differential operator $[g \cdot \bigtriangledown]^k$ is

$$[g \cdot \bigtriangledown]^k = \sum_{\substack{j_1+\cdots+j_m=k \\ 0 \le j_1,\cdots,j_m \le k}} \frac{k!}{j_1! \cdots j_m!} g_1^{j_1} \cdots g_m^{j_m} \frac{\partial^k}{\partial x_1^{j_1} \cdots \partial x_m^{j_m}}. \tag{11}$$

The last (remainder) term in (10) can be quantitatively bounded over $0 < \zeta < 1$ using interval arithmetic or other methods to obtain an interval remainder bound R_f. The summation in (10) is a q-th order polynomial (truncated Taylor series) in $(x - x_0)$ which we denote by $p_f(x - x_0)$. A q-th order Taylor model $T_f = p_f + R_f$ for $f(x)$ then consists of the polynomial p_f and the interval remainder bound R_f and is denoted by $T_f = (p_f, R_f)$. Note that $f \in T_f$ for $x \in X$ and thus T_f encloses the range of f over X.

Taylor models of functions can also be formed by performing Taylor model operations. Arithmetic operations with Taylor models can be done using the remainder differential algebra (RDA)

described by Makino and Berz [20, 23]. Let T_f and T_g be the Taylor models of the functions $f(x)$ and $g(x)$, respectively, over the interval $x \in X$. For $f \pm g$,

$$f \pm g \in T_f \pm T_g = (p_f, R_f) \pm (p_g, R_g) = (p_f \pm p_g, R_f \pm R_g). \tag{12}$$

Thus a Taylor model of $f \pm g$ is given by

$$T_{f\pm g} = (p_{f\pm g}, R_{f\pm g}) = (p_f \pm p_g, R_f \pm R_g). \tag{13}$$

For the product $f \times g$,

$$f \times g \in (p_f, R_f) \times (p_g, R_g) \subseteq p_f \times p_g + p_f \times R_g + p_g \times R_f + R_f \times R_g. \tag{14}$$

Note that $p_f \times p_g$ is a polynomial of order $2q$. Since a q-th order polynomial is sought for the Taylor model of $f \times g$, this term is split $p_f \times p_g = p_{f\times g} + p_e$. Here the polynomial $p_{f\times g}$ contains all terms of order q or less, and p_e contains the higher order terms. A q-th order Taylor model for the product $f \times g$ can then be given by $T_{f\times g} = (p_{f\times g}, R_{f\times g})$, with

$$R_{f\times g} = B(p_e) + B(p_f) \times R_g + B(p_g) \times R_f + R_f \times R_g. \tag{15}$$

Here $B(p) = P(X - x_0)$ denotes an interval bound on the polynomial $p(x - x_0)$ over $x \in X$. Similarly, an interval bound on an overall Taylor model $T = (p, R)$ will be denoted by $B(T)$, and is computed by obtaining $B(p)$ and adding it to the remainder bound R; that is, $B(T) = B(p) + R$. The method we use to obtain the polynomial bounds is described below. In storing and operating on a Taylor model, only the coefficients of the polynomial part p are used, and these are point valued. However, when these coefficients are computed in floating point arithmetic, numerical errors may occur and they must be bounded. To do this in our current implementation of Taylor model arithmetic, we have used the "tallying variable" approach, as described by Makino and Berz [23]. This approach has been analyzed in detail by Revol et al. [33]. This results in an error bound on the floating point calculation of the coefficients in p being added to the interval remainder bound R.

Taylor models for the reciprocal operation, as well as the intrinsic functions (exponential, logarithm, square root, sine, cosine, etc.) can also be obtained [19, 20, 23]. Using these, together with the basic arithmetic operations defined above, it is possible to start with simple functions such as the constant function $f(x) = k$, for which $T_f = (k, [0, 0])$, and the identity function $f(x_i) = x_i$, for which $T_f = (x_{i0} + (x_i - x_{i0}), [0, 0])$, and then to compute Taylor models for very complicated functions. Altogether, it is possible to compute a Taylor model for any function that can be represented in a computer environment by simple operator overloading through RDA operations. It has been shown that, compared to other rigorous bounding methods, the Taylor model often yields sharper bounds for modest to complicated functional dependencies [20, 21, 32]. A discussion of the uses and limitations of Taylor models has been given by Neumaier [32].

The range bounding of the interval polynomials $B(p) = P(X - x_0)$ is an important issue, which directly affects the performance of Taylor model methods. Unfortunately, exact range bounding of an interval polynomial is NP hard, and direct evaluation using interval arithmetic is very inefficient, often yielding only loose bounds. Thus, various bounding schemes [24, 32] have been used, mostly focused on exact bounding of the dominant parts of P, i.e., the first- and second-order terms. However, exact bounding of a general interval quadratic is also computationally expensive (in the worst case, exponential in the number of variables m). Thus, we have adopted here a very

simple compromise approach, in which only the first-order and the diagonal second-order terms are considered for exact bounding, and other terms are evaluated directly. That is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} \right)^2 + b_i (X_i - x_{i0}) \right] + Q, \tag{16}$$

where Q is the interval bound of all other terms, and is obtained by direct evaluation with interval arithmetic. In (16), since X_i occurs twice, there exists a dependency problem. For $|a_i| \geq \omega$, where ω is a small positive number, we can rearrange Eq. 16 such that each X_i occurs only once; that is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right] + Q. \tag{17}$$

In this way, the dependence problem in bounding the interval polynomial is alleviated so that a sharper bound can be obtained. If $|a_i| < \omega$, direct evaluation will be used instead.

4 Validated Parametric ODE Solver

In traditional interval Taylor series (ITS) methods for IVPs for ODEs, each integration step consists of two phases: 1. Validating existence and uniqueness; 2. Computing a tighter enclosure. The traditional methods are focused on handling interval-valued uncertainties in the initial conditions, and do not explicitly account for dependence on a parameter vector. Thus, in problems with interval-valued parameters, the interval dependency problem can quickly cause the size of the enclosure to become unacceptable, perhaps leading to premature termination of the integration process. As noted above, by treating interval parameters as additional state variables [16], tighter enclosures may be obtained using traditional methods. However, the increase in the number of state variables can result in a significant increase in the computational expense, due to, for example, the matrix factorizations used to control wrapping. In this section, we present a new method for the validated solution of the parametric ODE system defined by (1). A two-phase approach is used in which the dependence of $y'(t) = f(y, \theta)$ on t is handled using ITS methods. However, in the enclosure tightening phase, the dependence on the parameter vector θ is handled using Taylor models in terms of the parameters. Interval-valued uncertainties in the initial conditions are handled in the same way. This allows for efficient control of the interval dependency problem, and also leads to a new method for dealing with the wrapping effect. The technique for addressing the wrapping effect is based on a new type of Taylor model in which the enclosure of the remainder is an n-dimensional parallelepiped, rather than an interval.

4.1 Validating existence and uniqueness

In phase 1, we apply the traditional method to the parametric ODEs by using an interval Taylor series with respect to time. The approach used is essentially the same as in the traditional method, except that parameter dependence is explicitly accounted for. Assume that at t_j we have an enclosure Y_j of $y(t_j; t_0, Y_0, \Theta)$. The goal is to find a step size $h_j = t_{j+1} - t_j > 0$ and an a priori enclosure (coarse enclosure) $\widetilde{Y}_j$ of the solution such that a unique solution $y(t)$ is guaranteed to exist for all $t \in [t_j, t_{j+1}]$, all $y_j \in Y_j$, all $\theta \in \Theta$, and $y(t; t_j, Y_j, \Theta) \subseteq \widetilde{Y}_j$.

Using the Picard-Lindelöf operator and the Banach fixed-point theorem [5], it can be shown that if h_j and $Y_j \subseteq \widetilde{Y}_j^0$ satisfy

$$\widetilde{Y}_j = Y_j + [0, h_j]F(\widetilde{Y}_j^0, \Theta) \subseteq \widetilde{Y}_j^0, \tag{18}$$

then there is a unique solution $y(t; t_j, y_j, \theta)$ that satisfies $y(t; t_j, y_j, \theta) \in \widetilde{Y}_j$ for all $t \in [t_j, t_{j+1}]$, all $y_j \in Y_j$, all $\theta \in \Theta$. The method based on (18) is referred as a first-order (or constant) enclosure method. While an h_j for which (18) can be satisfied can always be found, this step size may be very small. Thus, high-order enclosure methods are commonly used that can enable larger step sizes by using polynomial enclosures [17] or more Taylor series terms [25, 28] in the sum in (18). Following the latter method, we determine h_j and $\widetilde{Y}_j$ such that

$$\widetilde{Y}_j = \sum_{i=0}^{k-1} [0, h_j]^i F^{[i]}(Y_j, \Theta) + [0, h_j]^k F^{[k]}(\widetilde{Y}_j^0, \Theta) \subseteq \widetilde{Y}_j^0. \tag{19}$$

Following the approach of Corliss and Rihm [4, Theorem 3], it can be shown that if (19) is satisfied, then there exists a unique solution $y(t; t_j, y_j, \theta) \in \widetilde{Y}_j$ for all $t \in [t_j, t_{j+1}]$, all $\theta \in \Theta$ and all $y_j \in Y_j$. The approach used to implement phase 1 is similar to that used in VNODE [26].

4.2 Computing a tighter enclosure

In phase 2, we compute a tighter enclosure $Y_{j+1} \subseteq \widetilde{Y}_j$ such that $y(t_{j+1}; t_0, Y_0, \Theta) \subseteq Y_{j+1}$. This will be done by using an ITS approach to compute a Taylor model $T_{y_{j+1}}$ of y_{j+1} in terms of the parameters and initial values, and then obtaining the enclosure Y_{j+1} from $Y_{j+1} = B(T_{y_{j+1}})$.

For the Taylor model computations, we begin by representing the interval initial values $y_0 \in Y_0$ by the Taylor model T_{y_0}, with components

$$T_{y_{0i}} = (m(Y_{0i}) + (y_{0i} - m(Y_{0i})), [0, 0]), \quad i = 1, \cdots, n. \tag{20}$$

The interval parameters are represented by the Taylor model T_θ, with components

$$T_{\theta_i} = (m(\Theta_i) + (\theta_i - m(\Theta_i)), [0, 0]), \quad i = 1, \cdots, p. \tag{21}$$

We can now determine Taylor models $T_{f^{[i]}}$ of the ITS coefficients $f^{[i]}(y_j, \theta)$ by using RDA operations and (7) to compute $T_{f^{[i]}} = f^{[i]}(T_{y_j}, T_\theta)$. The polynomial part of $T_{f^{[i]}}$ is specified to be of order q, and is a function of the parameters θ and initial values y_0 ($p+n$ total variables). Thus, $T_{f^{[i]}} = T_{f^{[i]}}(y_0, \theta)$.

Now consider the interval Taylor series with respect to time

$$y_{j+1} = y_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(y_j, \theta) + h_j^k f^{[k]}(y; t_j, t_{j+1}, \theta), \tag{22}$$

where $y_j \in Y_j$, and $f^{[k]}(y; t_j, t_{j+1}, \theta)$ denotes $f^{[k]}$ with its l-th component ($l = 1, \ldots, n$) evaluated at $y(\xi_{j,l})$ for some $\xi_{j,l} \in [t_j, t_{j+1}]$, which can be enclosed by $F^{[k]}(\widetilde{Y}_j, \Theta)$. To get an enclosure Y_{j+1} from (22), enclosures of the ITS coefficients $f^{[i]}(y_j, \theta)$, $i = 1, \ldots, k-1$ are needed. One traditional approach for doing this is to evaluate these coefficients in interval arithmetic, that is

$f^{[i]}(y_j,\theta) \in F^{[i]}(Y_j,\Theta)$. In one variation of the approach proposed here, we instead use Taylor models, that is $f^{[i]}(y_j,\theta) \in T_{f^{[i]}}(y_0,\theta)$, $y_0 \in Y_0$, $\theta \in \Theta$. Now using this with (22) gives the enclosure

$$y_{j+1} \in T_{y_{j+1}} = T_{y_j} + \sum_{i=1}^{k-1} h_j^i T_{f^{[i]}} + Z_{j+1}, \tag{23}$$

for $y_0 \in Y_0$ and $\theta \in \Theta$, where $Z_{j+1} = h_j^k F^{[k]}(\widetilde{Y}_j,\Theta)$. $T_{y_{j+1}}$ is a Taylor model of y_{j+1} with a q-th order polynomial in terms of the uncertain quantities θ and y_0. Y_{j+1} is then determined from $Y_{j+1} = B(T_{y_{j+1}})$; that is, by bounding $T_{y_{j+1}}$ over $y_0 \in Y_0$ and $\theta \in \Theta$. The use of Taylor models, rather than interval arithmetic to bound the ITS coefficients has the advantage that overestimation due to interval dependency is generally reduced significantly.

If either of these methods, the traditional interval evaluation or the proposed Taylor model evaluation, is used for bounding the ITS coefficients, the result is a "naive" approach in which the enclosure size does not contract, even in cases for which the true solution contracts. For instance, from (13) and (23), it can be seen that $R_{y_{j+1}}$, the interval remainder bound in $T_{y_{j+1}}$, is determined by adding the previous remainder bound R_{y_j} to the remainder bound arising from the summation term and to the interval Z_{j+1}. Since the width of a sum of intervals is equal to the sum of the widths of the intervals summed, this means that $w(R_{y_{j+1}}) \geq w(R_{y_j})$. Thus, the interval remainder part of the Taylor model keeps growing as the integration proceeds. This difficulty can be illustrated by a simple logistic map problem

$$y' = \theta y(1-y), \quad y_0 = 0.5, \quad \theta \in \Theta = [4.9, 5.0]. \tag{24}$$

For any $\theta \in \Theta$, the solution to this problem is $y \to 1$ as $t \to +\infty$. Using the naive approach, as given by (23), and using a step size of $h = 0.1$ (this is the maximum step size, it may be reduced in phase 1) we obtained at $t = 2.7$ an enclosure for the solution of $[0.9999837, 1.0000134]$, which is close to the true solution range. However, as the integration proceeds, the enclosure of the solution becomes larger and does not contract as the true solution range does. For example, at $t = 4.495688$, the solution enclosure obtained is $[0.8415998, 1.1584002]$. Eventually, the integration breaks down at about $t = 4.77$. This example will be revisited below in Section 5.

To address this issue, an approach used in connection with traditional methods is to bound the ITS coefficients by applying the mean value theorem, with evaluation of the resulting expression in interval arithmetic. In the new method described here, we instead evaluate mean-value enclosures of the ITS coefficients $f^{[i]}(y_j,\theta)$, $i = 1,\ldots,k-1$, using Taylor models. Applying the mean-value theorem at some point $\widehat{y}_j \in Y_j$, one obtains

$$f^{[i]}(y_j,\theta) = f^{[i]}(\widehat{y}_j,\theta) + J(f^{[i]};y_j,\widehat{y}_j,\theta)(y_j - \widehat{y}_j), \tag{25}$$

where $J(f^{[i]};y_j,\widehat{y}_j,\theta)$ denotes the Jacobian of $f^{[i]}$ with its l-th row ($l = 1,\ldots,n$) evaluated at $y_j + \xi_{i,l}(\widehat{y}_j - y_j)$ for some $\xi_{i,l} \in [0,1]$, and can be enclosed by $J(f^{[i]};Y_j,\Theta)$, an interval extension of the Jacobian of $f^{[i]}$ over $y_j \in Y_j$ and $\theta \in \Theta$. To apply the mean-value theorem using Taylor models, we first re-express T_{y_j} so that it has a remainder bound $\widehat{R}_{y_j}$ with $m(\widehat{R}_{y_j}) = 0$. This can always be done by adjusting the constant term in the polynomial part; after adjustment, this "centered" polynomial is denoted as $\widehat{T}_{y_j}$. That is, $T_{y_j} = (\widehat{T}_{y_j}, \widehat{R}_{y_j})$, and $\widehat{R}_{y_j} = T_{y_j} - \widehat{T}_{y_j}$. Note that the polynomial $\widehat{T}_{y_j}$ is point valued and that it now represents a point, say $\widehat{y}_j$, in $B(T_{y_j}) = Y_j$. Now evaluating (25) at the point $\widehat{y}_j = \widehat{T}_{y_j}$ and using Taylor model arithmetic, we obtain

$$T_{f^{[i]}} = T_{\widehat{f}^{[i]}} + J(f^{[i]};T_{y_j},\widehat{T}_{y_j},T_\theta)(T_{y_j} - \widehat{T}_{y_j}) = T_{\widehat{f}^{[i]}} + J(f^{[i]};T_{y_j},\widehat{T}_{y_j},T_\theta)\widehat{R}_{y_j}. \tag{26}$$

Here $T_{\widehat{f}^{[i]}} = f^{[i]}(\widehat{T}_{y_j}, T_\theta) = T_{\widehat{f}^{[i]}}(y_0, \theta)$, a q-th order Taylor model in terms of θ and y_0, and $J(f^{[i]}; T_{y_j}, \widehat{T}_{y_j}, T_\theta)$ is enclosed by $J(f^{[i]}; Y_j, \Theta)$ since $T_{y_j} \subseteq B(T_{y_j}) = Y_j$ and $T_\theta \subseteq \Theta$. Thus, for $f^{[i]}(y_j, \theta)$, we have the Taylor model enclosure

$$f^{[i]}(y_j, \theta) \in T_{\widehat{f}^{[i]}}(y_0, \theta) + J(f^{[i]}; Y_j, \Theta)\widehat{R}_{y_j}, \tag{27}$$

for $y_0 \in Y_0$ and $\theta \in \Theta$. Now using (22), we obtain the enclosure

$$y_{j+1} \in T_{y_{j+1}} = \widehat{T}_{y_j} + \sum_{i=1}^{k-1} h_j^i T_{\widehat{f}^{[i]}} + Z_{j+1} + S_j \widehat{R}_{y_j}, \tag{28}$$

with

$$S_j = I + \sum_{i=1}^{k-1} h_j^i J(f^{[i]}; Y_j, \Theta), \tag{29}$$

and $y_0 \in Y_0$, $\theta \in \Theta$. Again $T_{y_{j+1}}$ is a Taylor model of y_{j+1} with a q-th order polynomial in terms of the uncertain quantities θ and y_0, and Y_{j+1} is determined by bounding $T_{y_{j+1}}$ over $y_0 \in Y_0$ and $\theta \in \Theta$. Note that, in using (28), $R_{y_{j+1}}$, the remainder bound in $T_{y_{j+1}}$, is no longer determined from the sum of R_{y_j} and other intervals, as in (23). Thus, it is possible for the interval remainder part of the Taylor model to contract as the integration proceeds.

It remains to address the wrapping effect [25]. This occurs because the set $y(t_j; t_0, Y_0, \Theta)$ of solutions that we seek to enclose is rarely an interval. When this set is wrapped in an interval, overestimation occurs. If interval enclosures are used to propagate solution ranges from one time step to the next, this overestimation is also propagated and can thus grow rapidly. In traditional interval methods for ODEs, a common approach for dealing with this issue is to propagate intermediate results not as intervals, but as n-dimensional parallelepipeds (parallelepiped method) or n-dimensional rotated rectangles (QR method [16]), with the latter generally providing a more stable and effective approach. In the new method described above, results are propagated using Taylor models, not intervals, which ameliorates the wrapping effect. Nevertheless, because the Taylor models used involve an interval remainder bound, the wrapping effect remains an issue. For propagation of Taylor models, Makino and Berz [22] describe a "preconditioning" scheme, as further illustrated by Neher et al. [30]. Preconditioning involves representing the enclosure of the solution by a composition of a "left" and a "right" Taylor model. Depending on how the linear part of the left Taylor model is chosen, analogs of either the parallelepiped or QR methods can be obtained. For control of the wrapping effect in the method proposed here, we will use a different approach. This involves introduction of a new type of Taylor model in which the remainder bound is not represented by an interval.

Using a standard Taylor model, with interval remainder bound, results are propagated in the method described above using $T_{y_j} = \widehat{T}_{y_j} + \widehat{R}_{y_j}$. Thus, solutions have the form $y_j = \widehat{T}_{y_j} + r_j$, $r_j \in \widehat{R}_{y_j}$. To address the wrapping effect, we propose to instead propagate intermediate results using Taylor models of the form $T_{y_j} = \widehat{T}_{y_j} + \mathcal{P}_j$, where $\mathcal{P}_j = \{A_j v_j \mid v_j \in V_j\}$, and $A_j \in \mathbb{R}^{n \times n}$ is a real and regular matrix. This new type of Taylor model consists of the centered polynomial $\widehat{T}_{y_j}$ and a remainder bound represented by the set $\mathcal{P}_j$, which in general is an n-dimensional parallelepiped. Thus, solutions are now propagated in the form

$$y_j = \widehat{T}_{y_j} + A_j v_j, \tag{30}$$

$v_j \in V_j$, with $A_0 = I$ and $V_0 = 0$.

Given the results of the j-th time step, as represented by $\widehat{T}_{y_j}$, A_j and V_j, the solution at the next time step can be bounded using (22) and (26), with $T_{y_j} - \widehat{T}_{y_j} = \mathcal{P}_j$. This gives

$$y_{j+1} \in T_{y_{j+1}} = \widehat{T}_{y_j} + \sum_{i=1}^{k-1} h_j^i T_{\widehat{f}^{[i]}} + Z_{j+1} + S_j \mathcal{P}_j, \tag{31}$$

with S_j as in (29) and $y_0 \in Y_0$, $\theta \in \Theta$. This result is now re-centered by defining

$$T_{U_{j+1}} = \widehat{T}_{y_j} + \sum_{i=1}^{k-1} h_j^i T_{\widehat{f}^{[i]}} + Z_{j+1} \tag{32}$$

and re-expressing $T_{U_{j+1}}$ so that it has a remainder bound $\widehat{R}_{U_{j+1}}$ with $m(\widehat{R}_{U_{j+1}}) = 0$ and polynomial part $\widehat{T}_{U_{j+1}}$. Thus, from (31) and (32),

$$y_{j+1} \in T_{y_{j+1}} = \widehat{T}_{U_{j+1}} + \widehat{R}_{U_{j+1}} + S_j \mathcal{P}_j = \widehat{T}_{U_{j+1}} + \widehat{R}_{U_{j+1}} + S_j \{A_j v_j \mid v_j \in V_j\}. \tag{33}$$

Thus, solutions have the form

$$y_{j+1} = \widehat{T}_{U_{j+1}} + r_{j+1} + \breve{S}_j A_j v_j, \tag{34}$$

$\breve{S}_j \in S_j$, $r_{j+1} \in \widehat{R}_{U_{j+1}}$, $v_j \in V_j$. For propagation to the next time step, this result is put in the form (30) by setting

$$\widehat{T}_{y_{j+1}} = \widehat{T}_{U_{j+1}}, \tag{35}$$

and

$$A_{j+1} v_{j+1} \in \{r_{j+1} + \breve{S}_j A_j v_j | \breve{S}_j \in S_j, r_{j+1} \in \widehat{R}_{U_{j+1}}, v_j \in V_j\} \subseteq \widehat{R}_{U_{j+1}} + S_j A_j V_j. \tag{36}$$

Thus,

$$v_{j+1} \in V_{j+1} = A_{j+1}^{-1} \widehat{R}_{U_{j+1}} + \left(A_{j+1}^{-1} S_j A_j\right) V_j. \tag{37}$$

One choice for A_{j+1} is $A_{j+1} = m(S_j A_j)$, analogous to the traditional parallelepiped method. Another choice is $A_{j+1} = Q_j$, where Q_j is the orthogonal matrix obtained from $m(S_j A_j) = Q_j R_j$, the QR factorization of $m(S_j A_j)$. This is analogous to the traditional QR method. In this case, the set $\mathcal{P}_j$ in the modified Taylor model becomes a rotated n-dimensional rectangle. Usually the QR method provides better stability since, in the parallelepiped method, A_{j+1} may become ill conditioned, and in fact is not even guaranteed to be regular. If the QR factorization is performed with appropriate column permutations, then an improved enclosure can generally be obtained [16, 28]. Our implementation of QR is similar to that used in VNODE in this regard. Finally we obtain the next Taylor model as $T_{y_{j+1}} = \widehat{T}_{y_{j+1}} + \mathcal{P}_{j+1}$, $\mathcal{P}_{j+1} = \{A_{j+1} v_{j+1} | v_{j+1} \in V_{j+1}\}$. To obtain Y_{j+1}, this Taylor model is bounded over $y_0 \in Y_0$ and $\theta \in \Theta$. That is, $Y_{j+1} = B(T_{y_{j+1}}) = B(\widehat{T}_{y_{j+1}}) + A_{j+1} V_{j+1}$. It should be emphasized that when the traditional QR method is used, as in VNODE, the resulting enclosure is used directly for propagation to the next time step. However, when QR is used here to control wrapping, the resulting enclosure becomes part of a modified Taylor model, and this Taylor model is used for propagation to the next time step.

A summary of the new method for computing a tighter enclosure (phase 2) is given in Algorithm 1, for the case in which the QR method is used for control of the wrapping effect. This represents a completely new approach for phase 2. The method used for phase 1 is essentially the same as in the traditional approach implemented in VNODE, but extended to the case of interval-valued parameters.

Algorithm 1

Initialize: $A_0 = I$, $V_0 = 0$, $\widehat{T}_{y_0} = m(Y_0) + (y_0 - m(Y_0))$
Input: A_j, V_j, $\widehat{T}_{y_j}$, h_j (from phase 1), $\widetilde{Y}_j$ (from phase 1), Y_j, Θ
Compute:
$Z_{j+1} = h_j^k F^{[k]}(\widetilde{Y}_j, \Theta)$
$T_{U_{j+1}} = \widehat{T}_{y_j} + \sum_{i=0}^{k-1} h_j^i T_{\widehat{f}^{[i]}} + Z_{j+1}$
$S_j = I + \sum_{i=1}^{k-1} h_j^i J(f^{[i]}; Y_j, \Theta)$
$Q_j R_j = m(S_j A_j)$ (QR factorization with column permutations)
$A_{j+1} = Q_j$
$(\widehat{T}_{U_{j+1}}, \widehat{R}_{U_{j+1}}) \Leftarrow T_{U_{j+1}}$, making $m(\widehat{R}_{U_{j+1}}) = 0$
$\widehat{T}_{y_{j+1}} = \widehat{T}_{U_{j+1}}$
$V_{j+1} = (A_{j+1}^{-1} S_j A_j) V_j + A_{j+1}^{-1} \widehat{R}_{U_{j+1}}$
$T_{y_{j+1}} = \widehat{T}_{y_{j+1}} + \mathcal{P}_{j+1}$, $\mathcal{P}_{j+1} = \{A_{j+1} v_{j+1} | v_{j+1} \in V_{j+1}\}$
$Y_{j+1} = B(T_{y_{j+1}})$
Output: A_{j+1}, V_{j+1}, $\widehat{T}_{y_{j+1}}$, $T_{y_{j+1}}$, Y_{j+1}

5 Numerical experiments

We now report experimental results of a C++ implementation of the method described above. This implementation is called VSPODE (Validating Solver for Parametric ODEs). The automatic differentiation packages FADBAD [1] and TADIFF [2], which are both implemented in C++, are used to generate Taylor coefficients for the ITS expansions. We use our own implementations of interval arithmetic and of Taylor model arithmetic, which are also written in C++. Tests were performed on a workstation running Linux with an Intel Pentium 4 3.2GHz CPU. The results for VSPODE were obtained using a $k = 17$ order interval Taylor QR method, and with a $q = 5$ order Taylor model. For purposes of comparison, as a representative of traditional methods, we use the popular VNODE package [26], with a $k = 17$ order interval Hermite-Obreschkoff QR method. VNODE was tested with both the interval parameters as direct inputs and with the interval parameters as additional state variables. As expected, tighter enclosures were obtained using the latter approach, allowing for longer integration times before failure; only these VNODE results are presented here.

The first three example problems are simple ones for which there is an analytic solution. This allows for the solution enclosures calculated by VSPODE and VNODE to be compared with the exact solution ranges. The fourth example is a predator-prey model (Lotka-Volterra), which is a frequently used problem for the numerical verification of ODE solvers. The next example is the Lorenz problem [18], a crude model of atmospheric dynamics. The final example is the double pendulum problem. No analytic solutions are available for the last three problems. In the first five examples, a constant step size was selected. However, if necessary, this step size may be reduced in phase 1 of the procedure. To avoid the repeated need to reduce the step size in phase 1, a variable step size procedure is also available in both VNODE and VSPODE which will attempt to automatically select an appropriate step size. This variable step size procedure was used in the last example. The step size adjustment procedures used in VSPODE are the same as used in VNODE.

Table 1: Results on linear scalar problem, showing final enclosures ($t_m = 1$).

Method	Enclosure	Width
Analytic	[0.006096746565515, 0.007446583070925]	0.001350
VSPODE	[0.006096718470982, 0.007446583071959]	0.001350
VNODE	[0.005987, 0.007489]	0.001502

5.1 Linear scalar problem

In this problem we integrate

$$y' = -\theta y, \quad y(0) = 1, \quad \theta \in 5 + [-0.1, 0.1] \tag{38}$$

from $t_0 = 0$ to $t_m = 1$. Eq. (38) has the analytic solution

$$y = e^{-\theta t}. \tag{39}$$

Numerical tests were carried out with a constant step size $h = 0.02$. Table 5.1 shows the interval enclosures and their widths at $t_m = 1$ that are calculated via the analytic solution, and by using VSPODE and VNODE. The VSPODE enclosure bounds the analytic solution enclosure very tightly, to five significant figures on the lower bound and nine on the upper bound (as indicated by the underscores in Table 5.1). Based on the ratio of the widths of the calculated enclosure to the true enclosure, the relative overestimation by VNODE is about 11.3% and by VSPODE about 0.002%. The CPU time needed was 0.012 s for VNODE and 0.013 s for VSPODE.

5.2 Nonlinear scalar problem

In this problem we integrate

$$y' = -\theta y^2, \quad y(0) = 1, \quad \theta \in 5 + [-0.1, 0.1] \tag{40}$$

from $t_0 = 0$ to $t_m = 1$. Eq. (40) has the analytic solution

$$y = \frac{1}{1 + \theta t}. \tag{41}$$

Numerical tests were carried out with a constant step size $h = 0.02$. Table 2 shows the interval enclosures and their widths at $t_m = 1$ that are calculated from the analytic solution and by using VSPODE and VNODE. The enclosure of VSPODE is again a very tight bound on the exact solution. The relative overestimation, again based on the ratio of the calculated to true enclosure widths, is about 7.9% using VNODE and about 0.0002% using VSPODE. The CPU time needed was 0.018 s for VNODE and 0.073 s for VSPODE.

5.3 Logistic map problem

For this example, we revisit the logistic map problem (24) introduced in Section 4, and integrate it from $t_0 = 0$ to $t_m = 10$. This problem has the analytic solution

$$y = \frac{1}{1 + e^{-\theta t}}. \tag{42}$$

Table 2: Results on nonlinear scalar problem, showing final enclosures ($t_m = 1$).

Method	Enclosure	Width
Analytic	[0.1639344262295, 0.1694915254238]	0.005557
VSPODE	[0.1639344133626, 0.1694915254241]	0.005557
VNODE	[0.163669, 0.169664]	0.005994

Table 3: Results on logistic map problem.

	Enclosure		
t	Analytic	VSPODE	VNODE
0.5	[0.9205614508, 0.9241418200]	[0.9205614506, 0.9275735172]	[0.9203340057, 0.9279496341]
1	[0.9926084586, 0.9933071491]	[0.9926084586, 0.9939402237]	[0.9925307789, 0.9940835192]
2	[0.9999445514, 0.9999546022]	[0.9999445514, 0.9999628341]	[0.9999424927, 0.9999667116]
4	[0.9999999969, 0.9999999980]	[0.9999999969, 0.9999999987]	[0.9999999964, 0.9999999995]
10	[0.9999999999, 1.0000000000]	[0.9999999999, 1.0000000001]	[0.9999999999, 1.0000000001]

Numerical tests were carried out with a constant step size $h = 0.1$. Table 3 gives results at several intermediate time steps, showing the interval enclosures obtained, with comparison to the analytic result. Both methods can successfully integrate this problem and achieve the desired results, in which the computed enclosures contract. VSPODE obtains slightly better enclosures than VNODE. For integration to the final time of $t_m = 10$, the CPU time needed was 0.039 s for VNODE and 0.150 s for VSPODE. On this problem and on the previous one, VNODE was about 4 times faster than VSPODE. In general, we would expect a single step in VSPODE to require more computation time than a single step in VNODE. This is due primarily to the overhead involved in using Taylor model arithmetic in VSPODE, as opposed to only interval arithmetic in VNODE.

5.4 Lotka-Volterra problem

The Lotka-Volterra problem involves a model of a predator-prey system. The model describes the population dynamics of two species, with biomasses represented by y_1 and y_2. This is a widely studied model in theoretical ecology, and is also frequently used in the study of ODE solvers. The Lotka-Volterra model equations, along with the selected initial conditions and parameter intervals for this problem are

$$\begin{aligned} y_1' &= \theta_1 y_1(1 - y_2), \quad y_1(0) = 1.2, \quad \theta_1 \in 3 + [-0.01, 0.01], \\ y_2' &= \theta_2 y_2(y_1 - 1), \quad y_2(0) = 1.1, \quad \theta_2 \in 1 + [-0.01, 0.01]. \end{aligned} \tag{43}$$

The system was integrated from $t_0 = 0$ to $t_m = 10$. Numerical tests were carried out with a constant step size $h = 0.1$. No analytic solution is available for (43).

The enclosures computed using VSPODE and VNODE are shown in Figure 1, with more detailed results given for selected times in Table 4. VSPODE clearly provides a better enclosure, with VNODE breaking down at $t = 8.99$ due to rapid growth of the enclosure. However, VNODE

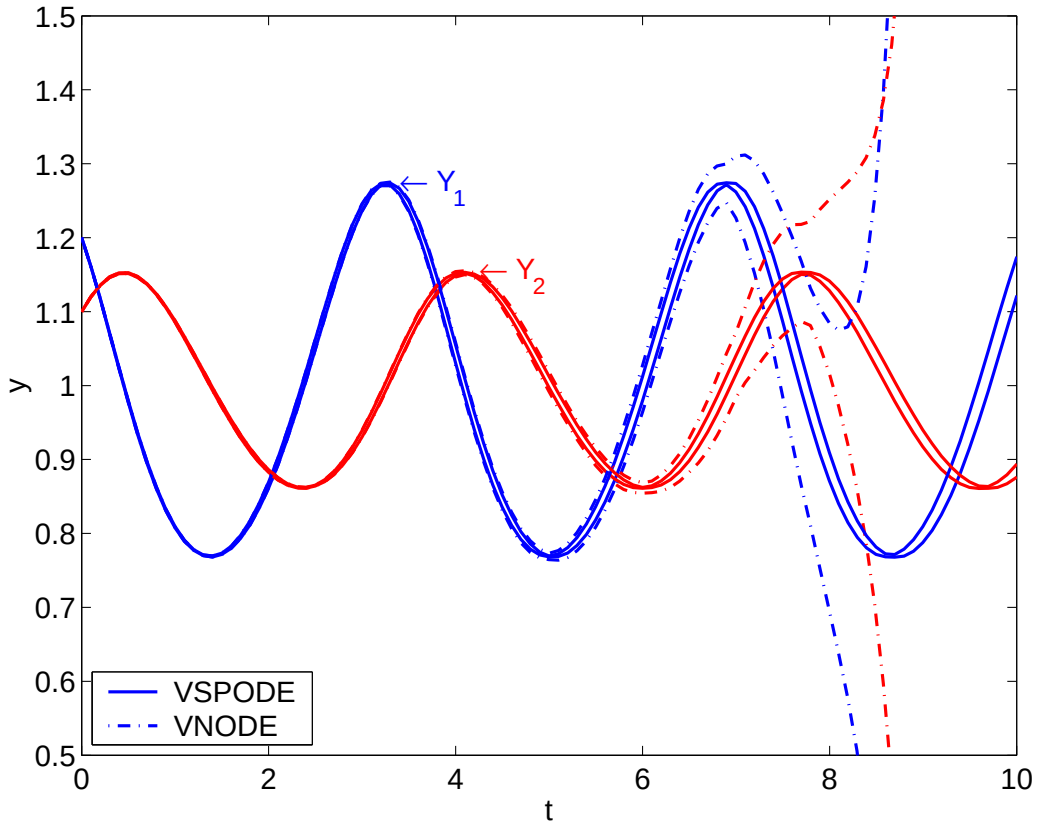

Figure 1: Solution enclosure of Lotka-Volterra equations computed using VSPODE and VNODE. Solid curves show the enclosures of the state variables as determined by VSPODE, and the dashed curves show the enclosures determined by VNODE.

is faster, and several runs of VNODE could be made with the same computational effort as one run of VSPODE. Thus, in order to allow VNODE multiple runs to solve the problem, we divided the parameter intervals into equal-sized sub-boxes and then used VNODE to determine the solution for each sub-box. The final solution enclosure is then the union of all the enclosures resulting from each parameter sub-box. Results showing the final solution enclosures ($t_m = 10$) and their widths, as determined using VSPODE, and VNODE with an increasing number of parameter sub-boxes, are given in Table 5. For example, VNODE-16 in Table 5 indicates the use of 16 parameter sub-boxes in VNODE. Even with 100 sub-boxes, the solution determined by VNODE is still wider than that obtained from a single calculation with VSPODE.

We also used this problem to study the effect of the choice of k, the order used for the interval Taylor series with respect to time, on the performance of VSPODE and VNODE. Automatic step size control was used for these tests. The results are shown in Table 6, which gives, for each method, the time through which the problem can be integrated before breakdown (due to rapid growth of the solution enclosure). Also shown are the total number of time steps and the total CPU time (s) taken to integrate up to the breakdown point. The trends observed in the results are the same for both VSPODE and VNODE. The use of higher order k clearly allows the use of fewer and larger time steps in phase 1. However, the larger time steps lead to a slight increase in the overestimation remaining after phase 2, and thus breakdown times tend to actually become slightly worse as k is increased.

Table 4: Results on Lotka-Volterra problem. If integration is continued past $t_m = 10$, VSPODE will eventually fail at $t = 31.8$.

	Enclosure	
t	VSPODE	VNODE
2	[0.8613, 0.8695]	[0.8612, 0.8696]
	[0.8825, 0.8861]	[0.8824, 0.8862]
4	[1.0303, 1.0549]	[1.0268, 1.0583]
	[1.1486, 1.1520]	[1.1466, 1.1541]
6	[0.9799, 1.0128]	[0.9644, 1.0282]
	[0.8610, 0.8624]	[0.8544, 0.8687]
8	[0.8702, 0.9083]	[0.6921, 1.0854]
	[1.1279, 1.1412]	[1.0105, 1.2574]
8.99		FAIL
10	[1.1208, 1.1737]	
	[0.8759, 0.8935]	

Table 5: Results on Lotka-Volterra problem, showing final enclosures ($t_m = 10$).

Method	Enclosure	Width	CPU time (s)
VSPODE	[1.120872, 1.173607]	0.052735	0.59
	[0.875994, 0.893472]	0.017478	
VNODE–16	[1.110859, 1.182814]	0.071955	1.42
	[0.872528, 0.898408]	0.025880	
VNODE–36	[1.116349, 1.177431]	0.061082	3.14
	[0.874923, 0.895612]	0.020689	
VNODE–64	[1.118150, 1.175692]	0.057542	5.59
	[0.875650, 0.894737]	0.019087	
VNODE–100	[1.118998, 1.174882]	0.055884	8.68
	[0.875975, 0.894337]	0.018362	

Table 6: Effect of k on breakdown time for integration of the Lotka-Volterra problem.

	VNODE			VSPODE		
k	breakdown time	steps	CPU time (s)	breakdown time	steps	CPU time (s)
4	2.57	184811	46.03	33.40	248993	157.64
5	9.71	8654	2.55	34.61	31020	25.70
6	9.71	8827	2.65	34.51	8021	8.53
7	9.65	1214	0.44	34.29	3208	4.42
8	9.65	1231	0.45	34.06	1666	2.76
9	9.43	407	0.18	33.72	1066	2.13
10	9.43	410	0.18	33.35	700	1.78
11	9.08	202	0.11	32.76	550	1.55
12	9.09	203	0.11	32.47	406	1.33
13	8.75	124	0.078	32.01	330	1.35
14	8.74	125	0.079	31.65	271	1.17
15	8.41	87	0.064	31.05	250	1.33
16	8.41	87	0.065	30.30	206	1.21
17	8.15	66	0.057	29.64	178	1.17

5.5 Lorenz Problem

The Lorenz ODE system is a crude model of atmospheric dynamics. We will integrate this system in the neighborhood of the classical parameter values $[10, 28, 8/3]$. The model equations are

$$\begin{aligned} y_1' &= \theta_1(y_2 - y_1), \quad y_1(0) = 10, \quad \theta_1 \in 10 + [-0.01, 0.01] \\ y_2' &= y_1(\theta_2 - y_3) - y_2, \quad y_2(0) = 10, \quad \theta_2 \in 28 + [-0.01, 0.01] \\ y_3' &= y_1 y_2 - \theta_3 y_3, \quad y_3(0) = 10, \quad \theta_3 \in 8/3 + [-0.01, 0.01]. \end{aligned} \tag{44}$$

The system was integrated from $t_0 = 0$ to $t_m = 2$. Numerical tests were done with a constant step size of $h = 0.01$. No analytic solution is available for (44).

The enclosures determined using VSPODE and VNODE are shown in Figure 2, with more detailed results for selected times given in Table 7. Again in this problem, VSPODE clearly provides a better enclosure, with VNODE breaking down at $t = 1.08$. Using a parameter interval subdivision approach with VNODE allows VNODE to solve the problem to $t = 2$, as shown in Table 8, which gives the final solution enclosures and their widths. Even compared to VNODE run on 1000 parameter sub-boxes, one run of VSPODE provides a sharper enclosure, and requires significantly less computation time.

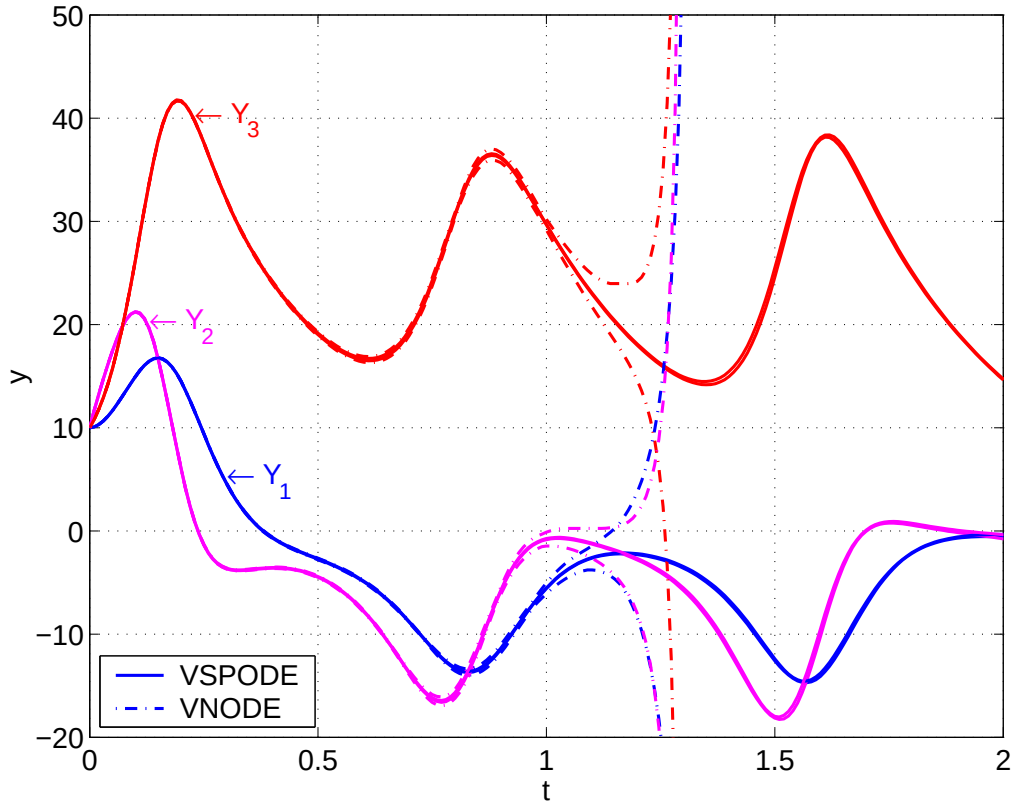

Figure 2: Solution enclosure of Lorenz system computed using VSPODE and VNODE. Solid curves show the enclosures of the state variables as determined by VSPODE (these upper and lower bounds are hard to distinguish on the scale used in this plot), and the dashed curves show the enclosures determined by VNODE.

Table 7: Results on Lorenz system problem. If integration is continued past $t_m = 2$, VSPODE will eventually break down at $t = 2.8$.

	Enclosure	
t	VSPODE	VNODE
0.5	[-2.7550, -2.6881]	[-2.8009, -2.6424]
	[-4.4973, -4.4247]	[-4.5511, -4.3715]
	[19.0057, 19.1629]	[18.8926, 19.2757]
1.0	[-5.5896, -5.5251]	[-6.1088, -5.0048]
	[-0.8471, -0.7433]	[-1.4974, -0.0929]
	[29.6447, 29.7035]	[29.1458, 30.2007]
1.25		FAIL
1.5	[-12.5807, -12.3754]	
	[-18.1466, -17.9698]	
	[24.5416, 25.4413]	
2.0	[-0.5821, -0.3423]	
	[-0.7696, -0.3693]	
	[14.63380, 14.7376]	

Table 8: Results on Lorenz system problem, showing final enclosures ($t_m = 2$).

Method	Enclosure	Width	CPU time (s)
VSPODE	[-0.582034, -0.342358]	0.2397	2.66
	[-0.769514, -0.369356]	0.4002	
	[14.633803, 14.737535]	0.1037	
VNODE–125	[-8.663337, 7.988072]	16.6514	33.7
	[-10.060513, 8.797512]	18.8580	
	[9.031894, 21.106684]	12.0748	
VNODE–512	[-0.920184, 0.041287]	0.9615	141.5
	[-1.321734, 0.245596]	1.5673	
	[14.352123, 15.010891]	0.6588	
VNODE–1000	[-0.770157, -0.136138]	0.6340	263.1
	[-1.077795, -0.036473]	1.0413	
	[14.502029, 14.869123]	0.3671	

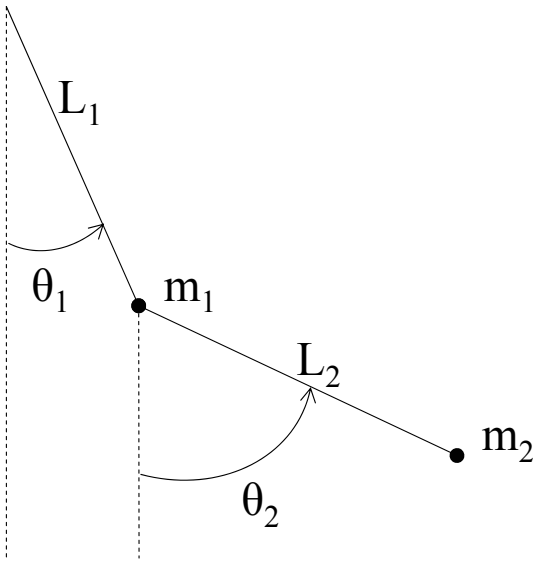

Figure 3: Schematic of double pendulum

5.6 Double Pendulum Problem

In this problem, we consider the motion of a double pendulum, as depicted in Fig. 3. This system can described by the nonlinear ODE system

$$\begin{aligned}
\dot{\theta}_1 &= \omega_1 \\
\dot{\theta}_2 &= \omega_2 \\
\dot{\omega}_1 &= \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g\sin(\theta_1-2\theta_2) - 2m_2\sin(\theta_1-\theta_2)\left[\omega_2^2 L_2 - \omega_1^2 L_1\cos(\theta_1-\theta_2)\right]}{L_1\left[2m_1+m_2-m_2\cos(2\theta_1-2\theta_2)\right]} \\
\dot{\omega}_2 &= \frac{2\sin(\theta_1-\theta_2)\left[\omega_1^2 L_1(m_1+m_2) + g(m_1+m_2)\cos\theta_1 + \omega_2^2 L_2 m_2\cos(\theta_1-\theta_2)\right]}{L_2\left[2m_1+m_2-m_2\cos(2\theta_1-2\theta_2)\right]},
\end{aligned} \tag{45}$$

where θ_1 and θ_2 are the angles of the pendulum rods (0 = vertical downwards, counter-clockwise is positive), and ω_1 and ω_2 are the angular velocities of the top and bottom rod, respectively. The mass parameters are set to $m_1 = m_2 = 1$ kg and the length parameters are set to $L_1 = L_2 = 1$ m. The initial conditions are $(\theta_1, \theta_2, \omega_1, \omega_2)_0 = (0, -0.25\pi, 0, 0)$. The parameter g is the local acceleration of gravity, which varies with latitude (greatest at the poles, lowest at the equator) and altitude. In this problem, we will treat g as an uncertain parameter in the interval $[9.79, 9.81]$ m/s^2. This corresponds roughly to the variation in the sea level value between 25° and 49° latitude (i.e., spanning the contiguous United States). The system was integrated from $t_0 = 0$ to $t_m = 8$. Numerical tests were done using the variable step size procedure (same in VSPODE as in VNODE). No analytic solution is available for (45).

The enclosures for the pendulum angles, θ_1 and θ_2, as computed by VSPODE and VNODE are shown in Figure 4, with VSPODE again providing significantly better results. More detailed results for all the state variables are given in Table 9 for selected times. As in the previous examples, use of parameter subdivision allows VNODE to integrate further, as shown in Table 10. With enough parameter subdivisions, VNODE can achieve comparable or better enclosures than VSPODE, but with an order of magnitude more computation time.

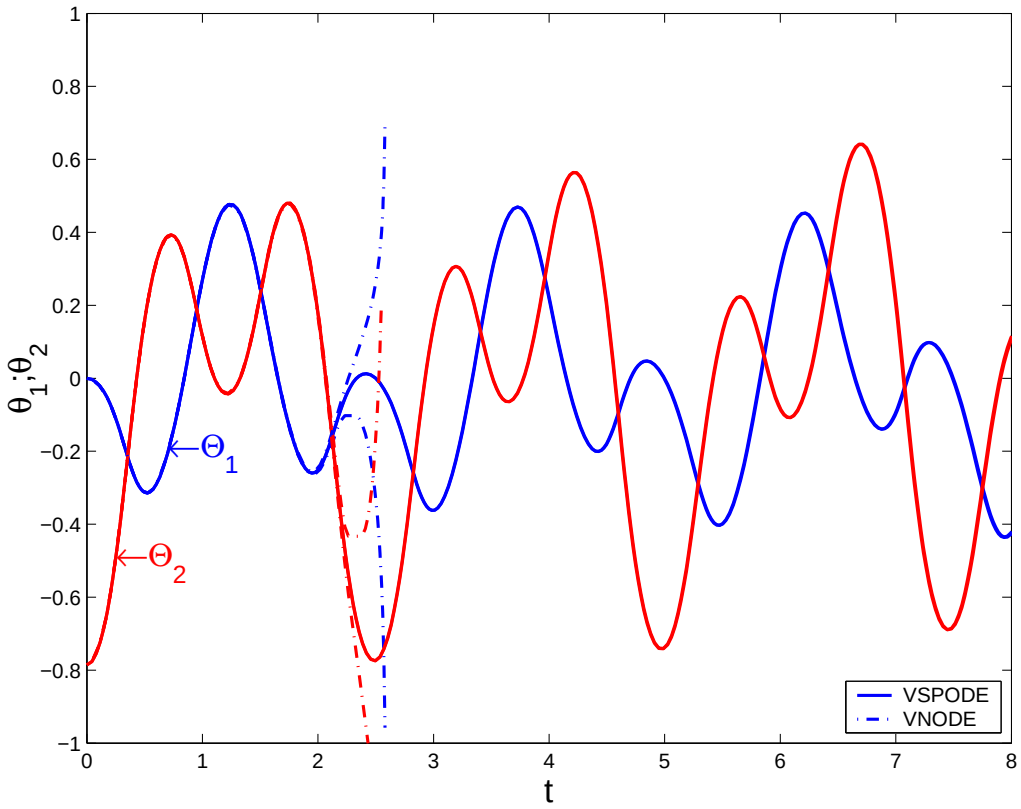

Figure 4: Solution enclosures of pendulum angles in the double pendulum problem, as computed using VSPODE and VNODE. Solid curves show the enclosures of the angles as determined by VSPODE, and the dashed curves show the enclosures determined by VNODE. The upper and lower bounds of the enclosures are hard to distinguish on the scale used in this plot.

6 Concluding Remarks

We have described a new method for obtaining validated solutions of initial value problems for ODEs with interval-valued parameters. Though we have focused on the case of parametric uncertainties, the method also accounts for interval-valued initial values. The method employs a traditional two-phase approach in which the dependence of $y'(t) = f(y, \theta)$ on t is handled using ITS methods. However, in the enclosure tightening phase, the dependence on parameters and initial values is handled using Taylor models. While there are other validated methods for ODEs that use Taylor models, they are determined and used here in a different way, and a new type of Taylor model is introduced for dealing with the wrapping effect. Numerical examples demonstrate that this approach provides a very effective way to determine tight enclosures of all possible solutions to parametric ODEs. The method presented, as implemented in VSPODE, is now being used in a number of practical applications, including simulation of bioreactors with parametric uncertainties [13], model-based fault diagnosis [14], deterministic global optimization for fitting time series data [10] and for optimal control [12], and state and parameter estimation for nonlinear, continuous-time systems [11]. Information about the availability of the VSPODE code can be obtained by contacting the authors.

Table 9: Results on double pendulum problem. State variable values are given in the order θ_1, θ_2, ω_1, ω_2. If integration is continued past $t_m = 8$, VSPODE will eventually break down at $t = 8.89$.

	Enclosure	
t	VSPODE	VNODE
2.0	[-0.2517, -0.2509]	[-0.2568, -0.2458]
	[0.1743, 0.1792]	[0.1693, 0.1842]
	[0.3533, 0.3689]	[0.3399, 0.3823]
	[-2.3539, -2.3339]	[-2.3731, -2.3148]
2.59		FAIL
4.0	[0.2160, 0.2224]	
	[0.3409, 0.3483]	
	[-1.5372, -1.5320]	
	[1.7901, 1.8020]	
6.0	[0.2927, 0.3014]	
	[-0.0856, -0.0818]	
	[1.3906, 1.4213]	
	[-0.6346, -0.5890]	
8.0	[-0.4261, -0.4213]	
	[0.1058, 0.1150]	
	[0.3924, 0.4578]	
	[0.7547, 0.8206]	

Table 10: Results on double pendulum problem, showing final enclosures ($t_m = 8$). State variable values are given in the order θ_1, θ_2, ω_1, ω_2.

Method	Enclosure	Width	CPU time (s)
VSPODE	[-0.426023, -0.421335]	0.004688	11.44
	[0.105861, 0.114998]	0.009137	
	[0.392437, 0.457779]	0.065342	
	[0.754772, 0.820508]	0.065736	
VNODE-100	[-0.432319, -0.414702]	0.017617	106.8
	[0.097406, 0.123391]	0.025985	
	[0.363618, 0.486597]	0.122979	
	[0.724180, 0.850733]	0.126553	
VNODE-150	[-0.425892, -0.421427]	0.004465	158.9
	[0.106347, 0.114483]	0.008136	
	[0.390422, 0.459761]	0.069339	
	[0.752335, 0.822922]	0.070587	
VNODE-200	[-0.425574, -0.421769]	0.003805	209.6
	[0.106909, 0.113937]	0.007028	
	[0.393237, 0.456966]	0.063729	
	[0.755262, 0.820009]	0.064747	

Acknowledgements

This work was supported in part by the State of Indiana 21st Century Research and Technology Fund under Grant #909010455, and by the Department of Energy under Grant DE-FG02-05CH11294. The authors also thank Professor George Corliss for reading an earlier draft of this paper and for making helpful comments and suggestions.

References

[1] C. Bendsten, 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, Lyngby, Denmark (1996).

[2] C. Bendsten, O. Stauning, TADIFF, a flexible C++ package for automatic differentiation using Taylor series, Technical Report 1997-x5-94, Technical University of Denmark, Lyngby, Denmark (1997).

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

[4] G. F. Corliss, R. Rihm, Validating an a priori enclosure using high-order Taylor series, in: G. Alefeld, A. Frommer, B. Lang (eds.), Scientific Computing and Validated Numerics: Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN'95), Akademie Verlag, Berlin, Germany (1995).

[5] P. Eijgenraam, The solution of initial value problems using interval arithmetic, Mathematical Centre Tracts No. 144, Stichting Mathematisch Centrum, Amsterdam, The Netherlands (1981).

[6] E. Hansen, G. W. Walster, Global Optimization Using Interval Analysis, Marcel Dekker, New York, NY (2004).

[7] M. Janssen, P. V. Hentenryck, Y. Deville, A constraint satisfaction approach for enclosing solutions to parametric ordinary differential equations, SIAM J. Num. Anal. 40 (2002) 1896–1939.

[8] L. Jaulin, M. Kieffer, O. Didrit, É. Walter, Applied Interval Analysis, Springer-Verlag, London, UK (2001).

[9] R. B. Kearfott, Rigorous Global Search: Continuous Problems, Kluwer Academic Publishers, Dordrecht, The Netherlands (1996).

[10] Y. Lin, M. A. Stadtherr, Deterministic global optimization for parameter estimation of dynamic systems, Ind. Eng. Chem. Res. (2006) in press.

[11] Y. Lin, M. A. Stadtherr, Guaranteed nonlinear state and parameter estimation for continuous-time systems, Automatica (2006) submitted.

[12] Y. Lin, M. A. Stadtherr, Deterministic global optimization of nonlinear dynamic systems, AIChE J. (2006) submitted.

[13] Y. Lin, M. A. Stadtherr, Validated solution of ODEs with parametric uncertainties, Computer-Aided Chem. Eng. 21 (2006) 167–172.

[14] Y. Lin, M. A. Stadtherr, Fault detection in continuous-time systems with uncertain parameters, submitted for presentation at 2007 American Control Conference, New York, NY (2007).

[15] R. J. Lohner, Enclosing the solutions of ordinary initial and boundary value problems, in: C. U. E. Kaucher, U. Kulisch (eds.), Computer Arithmetic: Scientific Computation and Programming Languages, Teubner, Stuttgart, Germany (1987).

[16] R. J. Lohner, Einschließung der lösung gewöhnlicher anfangs- und randwertaugfgaben und anwendungen, Ph.D. thesis, Universität Karlsruhe, Karlsruhe, Germany (1988).

[17] R. J. Lohner, Step size and order control in the verified solution of IVP with ODE's, in: SciCADE'95 International Conference on Scientific Computation and Differential Equations, Stanford, CA (1995).

[18] E. N. Lorenz, Deterministic non-periodic flow, J. Atmospheric Sci. 20 (1963) 130–141.

[19] K. Makino, Rigorous analysis of nonlinear motion in particle accelerators, Ph.D. thesis, Michigan State University, East Lansing, MI (1998).

[20] K. Makino, M. Berz, Remainder differential algebras and their applications, in: M. Berz, C. Bishof, G. Corliss, A. Griewank (eds.), Computational Differentiation: Techniques, Application, and Tools, SIAM, Philadelphia, PA (1996).

[21] K. Makino, M. Berz, Efficient control of the dependency problem based on Taylor model methods, Reliable Computing 5 (1999) 3–12.

[22] K. Makino, M. Berz, Suppression of the wrapping effect by Taylor model-based validated integrators, Technical Report MSUHEP 40910, Michigan State University, Lansing, MI (2003).

[23] K. Makino, M. Berz, Taylor models and other validated functional inclusion methods, Int. J. Pure Appl. Math. 4 (2003) 379–456.

[24] K. Makino, M. Berz, Taylor model range bounding schemes, in: Third International Workshop on Taylor Methods, Miami Beach (2004).

[25] R. E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ (1966).

[26] N. S. Nedialkov, Computing rigorous bounds on the solution of an initial value problems for an ordinary differential equation, Ph.D. thesis, University of Toronto, Toronto, Canada (1999).

[27] N. S. Nedialkov, K. R. Jackson, Some recent advances in validated methods for IVPs for ODEs, Appl. Numer. Math. 42 (2002) 269–284.

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

[29] N. S. Nedialkov, K. R. Jackson, J. D. Pryce, An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE, Reliable Computing 7 (2001) 449–465.

[30] M. Neher, K. R. Jackson, N. S. Nedialkov, On Taylor model based integration of ODEs, Technical Report, Department of Computer Science, University of Toronto, Toronto, Canada (2005).

[31] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, Cambridge, UK (1990).

[32] A. Neumaier, Taylor forms - use and limits, Reliable Computing 9 (2002) 43–79.

[33] N. Revol, K. Makino, M. Berz, Taylor models and floating point arithmetic: proof that arithmetic operations are bounded in COSY, J. Logic Algebr. Progr. 64 (2005) 135–154.

[34] A. B. Singer, Global dynamic optimization, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA (2004).

[35] A. B. Singer, P. I. Barton, Bounding the solutions of parameter dependent nonlinear ordinary differential equations, SIAM J. Sci. Comput. 27 (2006) 2167–2182.