Deterministic Global Optimization for Parameter Estimation of Dynamic Systems

Youdong Lin and Mark A. Stadtherr*

Department of Chemical and Biomolecular Engineering
University of Notre Dame, Notre Dame, IN 46556, USA

March 9, 2006

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

Abstract

A method is presented for deterministic global optimization in the estimation of parameters in models of dynamic systems. The method can be implemented as an ϵ-global algorithm, or, by use of the interval-Newton method, as an exact algorithm. In the latter case, the method provides a mathematically guaranteed and computationally validated global optimum in the goodness of fit function. A key feature of the method is the use of a new validated solver for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters, as well as on the first- and second-order sensitivities of the state variables with respect to the parameters. The computational efficiency of the method is demonstrated using several benchmark problems.

Keywords: Parameter estimation; Global optimization; Dynamic modeling; Interval analysis; Validated computing

1 Introduction

Parameter estimation is a key step in the development of mathematical models of physical phenomena, and is a well studied problem.[1] In many cases, especially in chemical engineering, the phenomena of interest are nonlinear in nature and are described by systems of ordinary differential equations (ODEs), or by differential-algebraic equation (DAE) systems. In general, there are two types of approaches to addressing the parameter estimation problem for such dynamic systems. In either approach, the objective is to minimize a weighted squared error between the observed values and those predicted by the model. The first approach uses integration routines to determine the values of the state variables, as well as the sensitivities, for a given set of parameter values. This is referred to as a *sequential* approach, since the solution of the differential system and the solution of the minimization problem are done sequentially. The other type of strategy is a *simultaneous* approach, in which discretization techniques, such as collocation, are used to transform the dynamic system into an algebraic system, resulting in a nonlinear programming (NLP) problem to which standard, or specially designed, solvers can be applied. A more detailed discussion of the optimization methods used in connection with both of these approaches is given by Esposito and Floudas.[2]

In the nonlinear parameter estimation of dynamic systems, it is not uncommon for there to be nonconvexities, leading to the important issue of multiplicity of local solutions.[2,3] Therefore, global optimization algorithms are needed to address this issue and find the globally optimal parameters. Stochastic searches[4] as well as deterministic methods[5] can be applied. The former class of methods, which essentially sample the feasible domain in various ways to locate the global optimum, increase the likelihood of finding the global optimum, but without a theoretical guarantee. Deterministic methods, on the other hand, can provide a theoretical guarantee of finding, in finite time, either the exact global optimum (e.g., interval-Newton methods[6]) or an ϵ-global optimum (e.g., αBB

methods[7,8]).

Esposito and Floudas[2] used the αBB approach for the solution of the minimization problem that arises in both the sequential and simultaneous approaches. They found that for systems with nonlinearities in the state variables, the simultaneous approach performed poorly compared to the sequential approach. The αBB approach uses convex underestimating functions in connection with a branch-and-bound strategy. A theoretical guarantee of attaining an ϵ-global solution is offered as long as rigorous underestimators are used, and this requires that sufficiently large values of α be used. However, the determination of proper values of α depends on the Hessian of the optimization problem, and, when the sequential approach is used, this matrix is not available in explicit functional form. Thus, Esposito and Floudas[2] did not use rigorous values of α in their implementation of the sequential approach, and so did not obtain a theoretical guarantee of global optimality. This issue is discussed in more detail by Papamichail and Adjiman.[9]

For rigorous determination of α, Chachuat and Latifi[10] have proposed two new approaches, one based on sensitivity analysis and the other on the use of adjoint variables. However, these procedures for determining first- and second-order derivatives of the necessary state-dependent functions appear to be quite costly. Papamichail and Adjiman[9,11] have presented a deterministic spatial branch-and-bound algorithm in which a new convex relaxation procedure is used. They achieved a rigorous convex relaxation of the dynamic information using either parameter independent or parameter dependent bounds on the solution of the dynamic system. Computational examples showed that the parameter independent bounds were not sufficiently tight, thus requiring many iterations to converge, and that the parameter dependent bounds, which are affine functions of the parameters, were tighter and thus required far fewer iterations. However, the computational cost of constructing such tight affine underestimators and overestimators appears to be high. The approaches given by both Chachuat and Latifi[10] and by Papamichail and Adjiman[9,11] provide a theoretical guarantee of

ϵ-global optimality. However, this is achieved at a high computational cost, as seen in the examples presented in section 6. Singer and Barton[12] have recently described a branch-and-bound approach using convex and concave relaxations for bounding the state trajectories. Using this approach, the cost of determining an ϵ-global optimum appears to be much more reasonable.

We present here a new deterministic global optimization approach for the parameter estimation of dynamic systems. This method is based on interval analysis and employs a type of sequential approach. A key feature of the method is the use of a new validated solver[13] for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters, as well as on the first- and second-order sensitivities of the state variables with respect to the parameters. The computational efficiency of this approach will be demonstrated through application to several benchmark problems.

The remainder of this paper is organized as follows. In section 2, the problem to be solved is defined mathematically. Section 3 provides a brief introduction to interval analysis and Taylor models, as well as a constraint propagation procedure on Taylor models. Section 4 reviews the new validated method[13] for parametric ODEs, which makes use of Taylor models. Section 5 then outlines the algorithmic procedure for solving the global optimization problem. Finally, in section 6 we present the results of some numerical experiments that demonstrate the effectiveness of the proposed approach for parameter estimation of dynamic systems.

2 Problem Definition

Let Z be the set of all state variables (components of $\boldsymbol{z}$), J be the set of states whose derivatives appear explicitly in the model ($J \subseteq Z$), and M be the set of fitted states ($M \subseteq Z$). In standard

error-in-variables form, the parameter estimation problem for a dynamic model can be written as

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{z}_\mu} \phi = & \sum_{m \in M} \sum_{\mu=1}^{r} (z_{\mu,m} - \bar{z}_{\mu,m})^2 \\ \text{s.}t. \quad & \dot{z}_j = g_j(\boldsymbol{z}, \boldsymbol{\theta}, t); \quad j \in J \\ & z_j(t_0) = z_{0,j}; \quad j \in J \\ & \mathbf{0} = \boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta}, t) \\ & z_{\mu,m} = z_m(t_\mu); \quad m \in M;\ \mu = 1, \ldots, r \\ & t \in [t_0, t_f]. \end{aligned} \tag{1}$$

Here $\boldsymbol{\theta}$ is the vector of parameters (length p) which are to be estimated; $\boldsymbol{z}_0 = (z_{0,j}; j \in J)$ is the vector (length $|J|$) of constant initial conditions; $\boldsymbol{z}_\mu = (z_{\mu,m}; m \in M)$ is the vector (length $|M|$) of fitted data variables and $\bar{\boldsymbol{z}}_\mu = (\bar{z}_{\mu,m}; m \in M)$ is the vector (length $|M|$) of measured values, both at $t = t_\mu$, the time associated with the μ-th data point; and r is the total number of data points. As stated, this is a DAE system with a vector (length $|J|$) of derivatives $\boldsymbol{g} = (g_j, j \in J)$ defining the differential constraints, and a vector $\boldsymbol{h}$ (length $|Z| - |J|$) defining the algebraic constraints. However, the DAE system is assumed to have an index of at most one. Therefore, it is possible to convert the DAE system into a set of ODEs, either by explicitly solving $\mathbf{0} = \boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta}, t)$ for the algebraic variables $z_i, i \notin J$, and substituting into $\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{\theta}, t)$ or through one differentiation of $\boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta}, t)$.[14] We will henceforth assume that the dynamic model in which parameters are to be estimated is in the form of a system of ODEs. We also assume that $\boldsymbol{g}$ is $(k-1)$-times continuously differentiable with respect to the state variables $\boldsymbol{z}$, and $(q+1)$-times continually differentiable with respect to the parameters $\boldsymbol{\theta}$. Here k is the order of the truncation error in the interval Taylor series (ITS) method to be used in the integration procedure (to be discussed in section 4), and q is the order of the Taylor model to be used to represent parameter dependence (to be discussed in section 3.2). When a typical sequential approach is used, an ODE solver is applied to the constraints with a given

set of parameter values, as determined by the optimization routine. This effectively eliminates $\boldsymbol{z}_\mu$, $\mu = 1, \ldots, r$, and leaves an unconstrained minimization in the parameters $\boldsymbol{\theta}$ only.

3 Interval Analysis and Taylor Models

3.1 Interval Analysis

A real interval X is defined as the set of real numbers lying between (and including) given upper and lower bounds; that is, $X = \left[\underline{X}, \overline{X}\right] = \left\{x \in \mathbb{R} \mid \underline{X} \leq x \leq \overline{X}\right\}$. Here an underline is used to indicate the lower bound of an interval and an overline is used to indicate the upper bound. A real interval vector $\boldsymbol{X} = (X_1, X_2, \cdots, X_n)^T$ has n real interval components and can be interpreted geometrically as an n-dimensional rectangle or box. Note that in this context uppercase quantities are intervals, and lowercase quantities or uppercase quantities with underline or overline are real numbers.

Basic arithmetic operations with intervals are defined by $X \text{ op } Y = \{x \text{ op } y \mid x \in X, y \in Y\}$, 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.[6,15–17] 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 $f(\boldsymbol{x})$, the interval extension $F(\boldsymbol{X})$ encloses all possible values of $f(\boldsymbol{x})$

for $\boldsymbol{x} \in \boldsymbol{X}$; that is, it encloses the range of $f(\boldsymbol{x})$ over $\boldsymbol{X}$. It is often computed by substituting the given interval $\boldsymbol{X}$ into the function $f(\boldsymbol{x})$ and then evaluating the function using interval arithmetic. This "natural" interval extension is often wider than the actual range of function values, though it always includes the actual range. For example, the natural interval extension of $f(x) = x/(x-1)$ over the interval $X = [2,3]$ is $F([2,3]) = [2,3]/([2,3]-1) = [2,3]/[1,2] = [1,3]$, while the true function range over this interval is $[1.5,2]$. This overestimation of the function range is due to the "dependency" problem, 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 $f(\boldsymbol{x})$ 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 $f(x) = x/(x-1) = 1 + 1/(x-1)$, and now $F([2,3]) = 1 + 1/([2,3]-1) = 1 + 1/[1,2] = 1 + [0.5,1] = [1.5,2]$, the true range.

In some situations, dependency issues can be avoided through the use of the dependent subtraction operation (also known as the cancellation operation). Assume that there is an interval S that depends additively on the interval A. The dependent subtraction operation is defined by $S \ominus A = [\underline{S} - \underline{A}, \overline{S} - \overline{A}]$. For example, let $A = [1,2]$, $B = [2,3]$, $C = [3,4]$ and $S = A+B+C = [6,9]$. Say that only S is stored and that later it is desired to compute $A + B$ by subtracting C from S. Using the standard subtraction operation yields $S - C = [6,9] - [3,4] = [2,6]$, which overestimates the true $A + B$. Using the dependent subtraction operation, which is allowable since S depends additively on C, yields $S \ominus C = [6,9] \ominus [3,4] = [3,5]$, which is the true $A + B$. For more gen-

eral situations, there are a variety of other approaches that can be used to try to tighten interval extensions,[6,15–17] including the use of Taylor models, as described in the next subsection.

3.2 Taylor Models

Makino and Berz[18] have described a remainder differential algebra (RDA) approach for bounding function ranges and control of the dependency problem of interval arithmetic.[19] This method employs high-order computational differentiation to express a function by a model consisting of a Taylor polynomial, usually a truncated Taylor series as shown below, and an interval remainder bound.

Consider a function $f : \boldsymbol{x} \in \boldsymbol{X} \subset \mathbb{R}^m \to \mathbb{R}$ that is $(q+1)$ times partially differentiable on $\boldsymbol{X}$ and let $\boldsymbol{x}_0 \in \boldsymbol{X}$. The Taylor theorem states that for each $\boldsymbol{x} \in \boldsymbol{X}$, there exists a $\zeta \in \mathbb{R}$ with $0 < \zeta < 1$ such that

$$f(\boldsymbol{x}) = \sum_{i=0}^{q} \frac{1}{i!} \left[(\boldsymbol{x} - \boldsymbol{x}_0) \cdot \bigtriangledown\right]^i f(\boldsymbol{x}_0) + \frac{1}{(q+1)!} \left[(\boldsymbol{x} - \boldsymbol{x}_0) \cdot \bigtriangledown\right]^{q+1} f\left[\boldsymbol{x}_0 + (\boldsymbol{x} - \boldsymbol{x}_0)\zeta\right], \tag{2}$$

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

$$[\boldsymbol{g} \cdot \bigtriangledown]^k = \sum_{\substack{j_1+\cdots+j_m=k \\ 0 \le j_1,\cdots,j_m \le k}} \frac{k!}{j_1! \cdots j_m!} g_1^{j_1} \cdots g_m^{j_m} \frac{\partial^k}{\partial x_1^{j_1} \cdots \partial x_m^{j_m}}. \tag{3}$$

The last (remainder) term in eq 2 can be quantitatively bounded over $0 < \zeta < 1$ using interval arithmetic or other methods to obtain an interval remainder bound. The Taylor model for $f(\boldsymbol{x})$ then consists of a q-th order polynomial in $(\boldsymbol{x} - \boldsymbol{x}_0)$, $p_f(\boldsymbol{x} - \boldsymbol{x}_0)$ (the summation in eq 2), and an interval remainder bound R_f. This Taylor model is denoted by $T_f = (p_f, R_f)$.

Arithmetic operations with Taylor models can be done using the RDA approach described by Makino and Berz.[18,20] Let T_f and T_g be the Taylor models (q-th order) of the functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ respectively over the interval $\boldsymbol{x} \in \boldsymbol{X}$. For $f \pm g$,

$$f \pm g \in T_f \pm T_g = (p_f, R_f) \pm (p_g, R_g) = (p_f \pm p_g, R_f \pm R_g). \tag{4}$$

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

$$T_{f\pm g} = (p_{f\pm g}, R_{f\pm g}) = (p_f \pm p_g, R_f \pm R_g). \tag{5}$$

For the the product $f \times g$,

$$f \times g \in (p_f, R_f) \times (p_g, R_g) \subseteq p_f \times p_g + p_f \times R_g + p_g \times R_f + R_f \times R_g. \tag{6}$$

Note that $p_f \times p_g$ is a polynomial of order $2q$. In order to be consistent with the q-th order polynomial in a Taylor model, this term is split into the sum of a polynomial $p_{f\times g}$ of up to q-th order, and an extra polynomial p_e containing the higher order terms. A 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

$$R_{f\times g} = B(p_e) + B(p_f) \times R_g + B(p_g) \times R_f + R_f \times R_g. \tag{7}$$

Here $B(p) = P(\boldsymbol{X} - \boldsymbol{x}_0)$ denotes an interval bound of the polynomial $p(\boldsymbol{x} - \boldsymbol{x}_0)$ over $\boldsymbol{x} \in \boldsymbol{X}$. 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, $B(T) = B(p) + R$. In storing and operating on a Taylor model, only the coefficients of the polynomial part $p(\boldsymbol{x} - \boldsymbol{x}_0)$ 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.[20] This approach has been analyzed in detail by Revol *et al.*[21] This results in an error bound on the floating point calculation of the coefficients in $p(\boldsymbol{x} - \boldsymbol{x}_0)$ being added to the interval remainder bound R.

The range bounding of the interval polynomials $B(p) = P(\boldsymbol{X} - \boldsymbol{x}_0)$ is an important issue, which directly affects the performance of the Taylor model. 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. Naturally, we focus on exact bounding of the dominant part, that is, the first- and second-order terms. However, exact bounding of a general interval quadratic is computationally expensive. Thus, we have adopted here a 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,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} \right)^2 + b_i (X_i - x_{i0}) \right] + Q, \tag{8}$$

where Q is the interval bound of all other terms, and is obtained by direct evaluation with interval arithmetic. In eq 8, 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 8 such that each X_i occurs only once; that is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right] + Q. \tag{9}$$

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

Taylor models for the reciprocal operation, as well as the intrinsic functions (exponential, logarithm, square root, sine, cosine, etc.) can also be obtained.[18,20,22] Using these, together with the basic arithmetic operations defined above, it is possible to start with simple functions such as the constant function $k(\boldsymbol{x}) = k$, for which $T_k = (k, [0, 0])$, and the identity function $i(x_i) = x_i, i = 1, \cdots, m$, for which $T_i = (x_{i0} + (x_i - x_{i0}), [0, 0])$, and to then 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.[18,19,23]

3.3 Constraint Propagation on Taylor Models

Partial information expressed by a constraint can be used to eliminate incompatible values from the domain of its variables. This domain reduction can then be propagated to all constraints on that variable, where it may be used to further reduce the domains of other variables. This process is known as constraint propagation. In this subsection, we show how to apply such a constraint propagation procedure using Taylor models, for both inequality and equality constraints.

Let T_c be the Taylor model of the function $c(\boldsymbol{x})$ over the interval $\boldsymbol{x} \in \boldsymbol{X}$, and say the constraint $c(\boldsymbol{x}) \leq 0$ needs to be satisfied. In the constraint propagation procedure (CPP) described here, $B(T_c)$ is determined and then there are three possible outcomes: 1. If $\underline{B(T_c)} > 0$, then no $\boldsymbol{x} \in \boldsymbol{X}$ will ever satisfy the constraint; thus, the CPP can be stopped and $\boldsymbol{X}$ discarded. 2. If $\overline{B(T_c)} \leq 0$, then every $\boldsymbol{x} \in \boldsymbol{X}$ will always satisfy the constraint; thus $\boldsymbol{X}$ cannot be reduced and the CPP can be stopped. 3. If neither of previous two cases occur, then part of the interval $\boldsymbol{X}$ may be eliminated; thus the CPP continues, using an approach based on the range bounding strategy for Taylor models described above.

For some $i \in \{1, 2, \ldots, m\}$, let a_i and b_i be the polynomial coefficients of the terms $(x_i - x_{i0})^2$ and $(x_i - x_{i0})$ of T_c, respectively. Note that, $x_{i0} \in X_i$ and is usually the midpoint $x_{i0} = m(X_i)$; the value of x_{i0} will not change during the CPP. For $|a_i| \geq \omega$, the bounds on T_c can be expressed using eq 9 as

$$B(T_c) = B(p) + R = a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} + S_i, \tag{10}$$

where S_i is determined by dependent subtraction (see section 3) using

$$S_i = B(T_c) \ominus \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right]. \tag{11}$$

Now define the intervals $U_i = X_i - x_{i0} + \frac{b_i}{2a_i}$ and $V_i = \frac{b_i^2}{4a_i} - S_i$, so that $B(T_c) = a_i U_i^2 - V_i$. The goal is to identify and retain only the part of X_i that contains values of x_i for which it is possible

to satisfy $c(\boldsymbol{x}) \leq 0$. Since $B(T_c)$ bounds the range of $c(\boldsymbol{x})$ for $\boldsymbol{x} \in \boldsymbol{X}$, the constraint $c(\boldsymbol{x}) \leq 0$ will be satisfied if $B(T_c) \leq 0$. Thus, to identify bounds on the part of X_i that satisfies the constraint, we can use the condition

$$a_i U_i^2 \leq V_i. \tag{12}$$

Then, the set U_i that satisfies eq 12, can be determined to be

$$U_i = \begin{cases} \emptyset & \text{if } a_i > 0 \text{ and } \overline{V_i} < 0 \\ \left[-\sqrt{\frac{\overline{V_i}}{a_i}}, \sqrt{\frac{\overline{V_i}}{a_i}}\right] & \text{if } a_i > 0 \text{ and } \overline{V_i} \geq 0 \\ [-\infty, \infty] & \text{if } a_i < 0 \text{ and } \overline{V_i} \geq 0 \\ \left[-\infty, -\sqrt{\frac{\overline{V_i}}{a_i}}\right] \cup \left[\sqrt{\frac{\overline{V_i}}{a_i}}, \infty\right] & \text{if } a_i < 0 \text{ and } \overline{V_i} < 0 \end{cases}. \tag{13}$$

The part of X_i to be retained is then $X_i = X_i \cap \left(U_i + x_{i0} - \frac{b_i}{2a_i}\right)$.

If $|a_i| < \omega$, then eq 9 cannot be used, but eq 8 can. Following a procedure similar to that used above, we now define $U_i = X_i - x_{i0}$ and $V_i = B(T_c) \ominus b_i(X_i - x_{i0})$. To identify bounds on the part of X_i that satisfies the constraint, we can now use the condition

$$b_i U_i \leq V_i. \tag{14}$$

Then, the set U_i that satisfies eq 14, can be determined to be

$$U_i = \begin{cases} \left[-\infty, \frac{\overline{V_i}}{b_i}\right] & \text{if } b_i > 0 \\ \left[\frac{\overline{V_i}}{b_i}, \infty\right] & \text{if } b_i < 0, \end{cases} \tag{15}$$

where it is assumed that $|b_i| \geq \omega$. The part of X_i to be retained is then $X_i = X_i \cap (U_i + x_{i0})$. If both $|a_i|$ and $|b_i|$ are less than ω, then no CPP will be applied on X_i.

For the equality constraint $c(\boldsymbol{x}) = 0$, the procedure is similar to but simpler than for the inequality case. If $|a_i| \geq \omega$, eq 12 becomes

$$a_i U_i^2 = V_i. \tag{16}$$

Then, with $W_i = V_i/a_i$, it follows that

$$U_i = \begin{cases} \emptyset & \text{if } \overline{W_i} < 0 \\ \left[-\sqrt{\overline{W_i}}, \sqrt{\overline{W_i}}\right] & \text{if } \underline{W_i} \le 0 \le \overline{W_i} \\ -\sqrt{W_i} \cup \sqrt{W_i} & \text{if } \underline{W_i} > 0 \end{cases} . \tag{17}$$

Thus, the part of X_i retained is $X_i = X_i \cap \left(U_i + x_{i0} - \frac{b_i}{2a_i}\right)$. If $|a_i| < \omega$ and $|b_i| \ge \omega$, eq 14 becomes $b_i U_i = V_i$, that is, $U_i = V_i/b_i$. Thus, in this case, the part of X_i retained is $X_i = X_i \cap (U_i + x_{i0})$.

The overall CPP is implemented by beginning with $i = 1$ and proceeding component by component. If, for any i, the result $X_i = \emptyset$ is obtained, then no $\boldsymbol{x} \in \boldsymbol{X}$ can satisfy the constraint; thus, $\boldsymbol{X}$ can be discarded and the CPP stopped. Otherwise the CPP proceeds until all components of $\boldsymbol{X}$ have been updated. Note that, in principle, each time an improved (smaller) X_i is found, it could be used in computing S_i for subsequent components of $\boldsymbol{X}$. However, this requires recomputing the bound $B(T_c)$, which, for the functions $c(\boldsymbol{x})$ that will be of interest here, is expensive. Thus, the CPP for each component is done using the bounds $B(T_c)$ computed from the original $\boldsymbol{X}$.

4 Validated Solution of Parametric ODEs

When a traditional sequential approach is applied to the parameter estimation problem, the objective function ϕ is evaluated, for a given value of $\boldsymbol{\theta}$, by applying an ODE solver to the constraints to eliminate $\boldsymbol{z}_\mu$, $\mu = 1, \ldots, r$. In the global optimization algorithm described here, we will use a sequential approach based on interval analysis. This approach requires the evaluation of bounds on ϕ, given some parameter interval $\boldsymbol{\Theta}$. Thus, we need an ODE solver that can compute bounds on $\boldsymbol{z}_\mu$, $\mu = 1, \ldots, r$, for the case in which the parameters are interval valued. Interval methods[24] (also called validated methods) for ODEs provide a natural approach for computing the desired enclosures of the state variables at the times t_μ, $\mu = 1, \ldots, r$, corresponding to the given data

points.

Validated methods for ODEs not only can determine guaranteed bounds on the state variables, but can also verify that a unique solution to the problem exists. Traditional interval methods usually consist of two processes applied at each integration step.[24] In the first process, existence and uniqueness of the solution are proven using the Picard-Lindelöf operator and the Banach fixed point theorem, and a rough enclosure of the solution is computed. In the second process, a tighter enclosure of the solution is computed. 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. An excellent review of the traditional interval methods has been given by Nedialkov *et al.*[25] For addressing this problem, there are various packages available, including AWA,[26] VNODE[27] and COSY VI,[28] all of which consider uncertainties (interval valued) in initial values only. Recently, Lin and Stadtherr[13] have proposed a method for efficiently determining validated solutions of ODEs with interval-valued parameters. The method makes use, in a novel way, of the Taylor model approach[18–20] to deal with the dependency problem on the uncertain variables (parameters and initial values). In the context of parameter estimation, the initial values are assumed to be known exactly, and so only the parameters are interval valued. We will summarize here the basic ideas of the method used. Additional details are given by Lin and Stadtherr.[13]

Consider the following parametric ODE system, with state variables denoted by $\boldsymbol{y}$:

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{\theta}), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{\theta} \in \boldsymbol{\Theta}, \tag{18}$$

where $t \in [t_0, t_m]$ for some $t_m > t_0$. Note that a parameter interval $\boldsymbol{\Theta}$ has been specified, and that it is desired to determine a validated enclosure of all possible solutions to this initial value problem. Also note that nonautonomous (time dependent) problems can be converted to the autonomous form given in eq 18. We denote by $\boldsymbol{y}(t; t_j, \boldsymbol{Y}_j, \boldsymbol{\Theta})$ the set of solutions $\boldsymbol{y}(t; t_j, \boldsymbol{Y}_j, \boldsymbol{\Theta}) = \left\{ \boldsymbol{y}(t; t_j, \boldsymbol{y}_j, \boldsymbol{\theta}) \mid \boldsymbol{y}_j \in \boldsymbol{Y}_j, \boldsymbol{\theta} \in \boldsymbol{\Theta} \right\}$, where $\boldsymbol{y}(t; t_j, \boldsymbol{y}_j, \boldsymbol{\theta})$ denotes a solution of $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{\theta})$ for the

initial condition $\boldsymbol{y} = \boldsymbol{y}_j$ at $t = t_j$. We will describe a method for determining enclosures $\boldsymbol{Y}_j$ of the state variables at each time step $j = 1, \ldots, m$, such that $\boldsymbol{y}(t_j; t_0, \boldsymbol{y}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{Y}_j$.

Assume that at t_j we have an enclosure $\boldsymbol{Y}_j$ of $\boldsymbol{y}(t_j; t_0, \boldsymbol{y}_0, \boldsymbol{\Theta})$, and that we want to carry out an integration step to compute the next enclosure $\boldsymbol{Y}_{j+1}$. Then, in the first phase of the method, the goal is to find a step size $h_j = t_{j+1} - t_j > 0$ and a prior enclosure $\tilde{\boldsymbol{Y}}_j$ of the solution such that a unique solution $\boldsymbol{y}(t; t_j, \boldsymbol{y}_j, \boldsymbol{\theta}) \in \tilde{\boldsymbol{Y}}_j$ is guaranteed to exist for all $t \in [t_j, t_{j+1}]$, any $\boldsymbol{y}_j \in \boldsymbol{Y}_j$, and any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. We apply the traditional interval method, with high order enclosure, to the parametric ODEs by using an interval Taylor series (ITS) with respect to time. That is, we determine h_j and $\tilde{\boldsymbol{Y}}_j$ such that for $\boldsymbol{Y}_j \subseteq \tilde{\boldsymbol{Y}}_j^0$,

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

Here k denotes the order of the Taylor expansion, and the coefficients $\boldsymbol{F}^{[i]}$ are interval extensions of the Taylor coefficients $\boldsymbol{f}^{[i]}$ of $\boldsymbol{y}(t)$ with respect to time, which can be obtained recursively in terms of $\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{\theta})$ by

$$\begin{aligned} \boldsymbol{f}^{[0]} &= \boldsymbol{y} \\ \boldsymbol{f}^{[1]} &= \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{\theta}) \\ \boldsymbol{f}^{[i]} &= \frac{1}{i} \left(\frac{\partial \boldsymbol{f}^{[i-1]}}{\partial \boldsymbol{y}} \boldsymbol{f} \right) (\boldsymbol{y}, \boldsymbol{\theta}), \qquad i \geq 2. \end{aligned} \tag{20}$$

Satisfaction of eq 19 demonstrates that there exists a unique solution $\boldsymbol{y}(t; t_j, \boldsymbol{y}_j, \boldsymbol{\theta}) \in \tilde{\boldsymbol{Y}}_j$ for all $t \in [t_j, t_{j+1}]$, any $\boldsymbol{y}_j \in \boldsymbol{Y}_j$, and any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

In phase 2, we compute a tighter enclosure $\boldsymbol{Y}_{j+1} \subseteq \tilde{\boldsymbol{Y}}_j$ such that $\boldsymbol{y}(t_{j+1}; t_0, \boldsymbol{y}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{Y}_{j+1}$. This will be done by using an ITS approach to compute a Taylor model $\boldsymbol{T}_{\boldsymbol{y}_{j+1}}$ of $\boldsymbol{y}_{j+1}$ in terms of the parameters, and then obtaining the enclosure $\boldsymbol{Y}_{j+1} = B(\boldsymbol{T}_{\boldsymbol{y}_{j+1}})$. For the Taylor model computations, we begin by representing the parameters by the Taylor model $\boldsymbol{T}_{\boldsymbol{\theta}}$ with components

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

where $m(\Theta_i)$ indicates the midpoint of the interval Θ_i. Then, we can determine Taylor models $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$ of the interval Taylor series coefficients $\boldsymbol{f}^{[i]}(\boldsymbol{y}_j, \boldsymbol{\theta})$ by using RDA operations to compute $\boldsymbol{T}_{\boldsymbol{f}^{[i]}} = \boldsymbol{f}^{[i]}(\boldsymbol{T}_{\boldsymbol{y}_j}, \boldsymbol{T}_{\boldsymbol{\theta}})$. Using an interval Taylor series for $\boldsymbol{y}_{j+1}$ with coefficients given by $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$, and incorporating a novel approach for using the mean value theorem on Taylor models, one can obtain a result for $\boldsymbol{T}_{\boldsymbol{y}_{j+1}}$ in terms of the parameters. To further control the wrapping effect,[24] a QR factorization approach is applied to the remainder bound. The algorithmic procedure for phase 2 is summarized in Algorithm 1, where $\boldsymbol{J}(\boldsymbol{f}^{[i]}; \boldsymbol{Y}_j, \boldsymbol{\Theta})$ denotes the interval extension of the Jacobian (with respect to the state variables) of $\boldsymbol{f}^{[i]}$ over $\boldsymbol{y}_j \in \boldsymbol{Y}_j$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The procedure begins with $\boldsymbol{V}_0 = \boldsymbol{0}$, $\hat{\boldsymbol{T}}_{\boldsymbol{y}_0} = (\boldsymbol{y}_0, [0, 0])$, and $A_0 = I$. Complete details of the computation of $\boldsymbol{T}_{\boldsymbol{y}_{j+1}}$ are given by Lin and Stadtherr.[13] If necessary, it may be possible to further tighten $\boldsymbol{Y}_{j+1}$ if one or more of the state variables has some known physical bounds that could be applied. The use of physical state bounds is an important feature of the method used by Singer and Barton.[12] We have not made use of any physical state bounds in the example problems given below.

An implementation of this approach, called VSPODE (Validating Solver for Parametric ODEs), has been developed and tested by Lin and Stadtherr,[13] who compared its performance with results obtained using the popular VNODE package.[27] For the test problems used, VSPODE provided tighter enclosures on the state variables than VNODE, and required significantly less computation time.

5 Global Optimization Procedure

The global optimization procedure described here uses an interval-Newton approach.[6,16] The algorithm can be thought of as a type of branch-and-bound method, with various strategies used for domain reduction. Since a sequential approach to parameter estimation is used here, the problem becomes the unconstrained minimization of $\phi(\boldsymbol{\theta})$. The interval-Newton (IN) method provides the

capability to find tight enclosures of all global minimizers of ϕ in some specified initial interval $\mathbf{\Theta}^{(0)}$, and to do so with *mathematical and computational certainty*. The IN method is basically an equation-solving method. In the context of unconstrained minimization, it is used to seek solutions of $\boldsymbol{f}(\boldsymbol{\theta}) = \nabla\phi(\boldsymbol{\theta}) = \mathbf{0}$, where $\nabla\phi(\boldsymbol{\theta})$ indicates the gradient of the objective function $\phi(\boldsymbol{\theta})$. Given some initial interval $\mathbf{\Theta}^{(0)}$ sufficiently large that the global minimum sought is in its interior, the IN algorithm is applied to a sequence of subintervals. At the k-th iteration, three steps, including the objective range test, function range test, and interval-Newton test, may be applied on the current subinterval $\mathbf{\Theta}^{(k)}$.

5.1 Objective Range Test

In this step, the Taylor model T_{ϕ_k}, of the objective function $\phi(\boldsymbol{\theta})$ over $\mathbf{\Theta}^{(k)}$ is computed. To do this, Taylor models of $\boldsymbol{z}_\mu$, the state variables at times t_μ, $\mu = 1, \ldots, r$, corresponding to the given data points, must first be determined. This is done using VSPODE, as described in section 4. An interval bound $B(T_{\phi_k})$ of T_{ϕ_k} is then obtained using the bounding procedure for Taylor models given in section 3.2. The final step in bounding each term in the sum of squares function $\phi(\boldsymbol{\theta})$ is actually done in two different ways. Each term in $\phi(\boldsymbol{\theta})$ is of the form $(z - \bar{z})^2 = (\Delta z)^2$. After $T_{\Delta z}$ has been obtained, then both $B(T_{\Delta z}^2)$ and $(B(T_{\Delta z}))^2$ are determined. The contribution to $B(T_{\phi_k})$ is then the intersection of these two results. This tends to tighten the bounds provided by $B(T_{\phi_k})$ and also guarantees that $\underline{B(T_{\phi_k})} \geq 0$.

The part of $\mathbf{\Theta}^{(k)}$ that can contain the global minimum must satisfy the constraint $\phi(\boldsymbol{\theta}) - \hat{\phi} \leq 0$, where $\hat{\phi}$ is a known upper bound on the global minimum, the initialization and update of which is discussed below. Thus the constraint propagation procedure (CPP) described in section 3.3 is now applied using this constraint. The first step of the CPP amounts to checking if the lower bound of T_{ϕ_k}, $\underline{B(T_{\phi_k})}$, is greater than $\hat{\phi}$. If so, then $\mathbf{\Theta}^{(k)}$ can be discarded because it cannot contain the

global minimum and need not be further tested. The second step of the CPP amounts to checking if the upper bound of T_{ϕ_k}, $\overline{B(T_{\phi_k})}$, is less than $\hat{\phi}$. If so, then all points in $\mathbf{\Theta}^{(k)}$ satisfy the constraint and the CPP can be stopped since no reduction in $\mathbf{\Theta}^{(k)}$ can be achieved. This also indicates, with certainty, that there is a point in $\mathbf{\Theta}^{(k)}$ that can be used to update $\hat{\phi}$. Thus, if $\overline{B(T_{\phi_k})} < \hat{\phi}$, a local optimization routine, starting at some point in $\mathbf{\Theta}^{(k)}$, is used to find a local minimum, which then provides an updated (smaller) $\hat{\phi}$, that is, a better upper bound on the global minimum. In our implementation, the midpoint of $\mathbf{\Theta}^{(k)}$ is used as the starting point for the local optimization. A new CPP is then started on $\mathbf{\Theta}^{(k)}$ using the updated value of $\hat{\phi}$. If neither of the previous two outcomes occurs, then the full CPP described in section 3.3 is applied to reduce $\mathbf{\Theta}^{(k)}$. If $\mathbf{\Theta}^{(k)}$ is sufficiently reduced (by more than 10% by volume), then new bounds $B(T_{\phi_k})$ are obtained, now over the smaller $\mathbf{\Theta}^{(k)}$, and a new CPP is started. Otherwise, the processing of $\mathbf{\Theta}^{(k)}$ continues with the function range test.

In this application, the objective function ϕ is a sum of squares function, and it can be accumulated as a series of partial sums computed at each data time t_μ, $\mu = 1, \ldots, r-1$, with the final sum ϕ determined only after integration to the final data time t_r. Thus, after integration through the s-th data time t_s $(s < r)$, we have computed the partial sum

$$\phi_s = \sum_{m \in M} \sum_{\mu=1}^{s} (z_{\mu,m} - \bar{z}_{\mu,m})^2. \tag{22}$$

Since each term in the sum is positive, $\phi_s \leq \phi_{s+1} \leq \phi$, $s = 1, \ldots, r-1$. Thus, for any $s = 1, \ldots, r-1$, a lower bound on ϕ_s is a valid lower bound on ϕ, and the bounds improve (increase) as s increases. However, an upper bound on ϕ_s is not a valid upper bound on ϕ. This means that we are able to augment the objective range test described above by applying a partial CPP after each data time t_s is reached in the integration process. At this point, the Taylor model of ϕ_s over $\mathbf{\Theta}^{(k)}$ is available. It is bounded to obtain $B(T_{\phi_{sk}})$ and a CPP is started. The only part of the CPP that cannot be done is the second step, which determines when to update $\hat{\phi}$, since this involves the upper bound

on the Taylor model of ϕ_s, which is not a valid upper bound on ϕ. Using this partial CPP at each data time may result in $\boldsymbol{\Theta}^{(k)}$ being eliminated without having to integrate all the way to t_r, or it may result in a larger reduction of $\boldsymbol{\Theta}^{(k)}$ once t_r is reached.

As with any type of procedure incorporating branch-and-bound, an important issue is how to initialize $\hat{\phi}$, the upper bound on the global minimum. There are many ways in which this can be done, and clearly, it is desirable to find a $\hat{\phi}$ that is as small as possible (i.e., the tightest possible upper bound). To initialize $\hat{\phi}$, we run p^2 local minimizations (p is the number of parameters to be estimated) using a local optimization routine from randomly chosen starting points, and then choose the smallest value of ϕ found to be the initial $\hat{\phi}$. For this purpose, we use the bound-constrained quasi-Newton method L-BFGS-B[29] as the local optimization routine, and DDASSL[14] as the integration routine.

5.2 Function Range Test

This test step is also known as the gradient test. Since the global minimum must be one of the stationary points of $\phi(\boldsymbol{\theta})$, it must be a solution of $\boldsymbol{f}(\boldsymbol{\theta}) = \nabla\phi(\boldsymbol{\theta}) = \mathbf{0}$. The Taylor model $\boldsymbol{T}_{\boldsymbol{f}_k}$, of $\boldsymbol{f}(\boldsymbol{\theta})$ over $\boldsymbol{\Theta}^{(k)}$ is computed. If there is any component of $B(\boldsymbol{T}_{\boldsymbol{f}_k})$ that does not include zero, then no solution of $\boldsymbol{f}(\boldsymbol{\theta}) = \mathbf{0}$ can exist in this interval. This interval can then be discarded. Note that we have assumed that the initial interval $\boldsymbol{\Theta}^{(0)}$ is sufficiently large that the global minimum will not be on its boundary, because an extremum on the boundary is in general not a stationary point. For situations in which such an assumption cannot be made, the "peeling" process described by Kearfott,[16] in which IN is applied to each of the lower dimensional subspaces that constitute the boundary of $\boldsymbol{\Theta}^{(0)}$, can be used.

Using $\boldsymbol{T}_{\boldsymbol{f}_k}$, a constraint propagation procedure can also be performed using each component of the stationarity constraint $\boldsymbol{f}(\boldsymbol{\theta}) = \mathbf{0}$. A CPP is done for each component of $\boldsymbol{\Theta}^{(k)}$, using each

component of the stationarity constraint, until all components of $\boldsymbol{\Theta}^{(k)}$ have been updated, or $\boldsymbol{\Theta}^{(k)}$ has been discarded. The procedure is repeated until no further improvement of $\boldsymbol{\Theta}^{(k)}$ can be made.

Note that in order to determine a Taylor model of $\boldsymbol{f}(\boldsymbol{\theta})$, it is necessary to obtain Taylor models for the first-order sensitivities $\boldsymbol{z}_{\theta_i} = \partial \boldsymbol{z}/\partial\theta_i$, $i = 1, \ldots, p$, at each t_μ, $\mu = 1, \ldots, r$. To do this, VSPODE is applied to integrate the first-order sensitivity equation,

$$\begin{aligned} \dot{\boldsymbol{z}}_{\theta_i} &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} \boldsymbol{z}_{\theta_i} + \frac{\partial \boldsymbol{g}}{\partial \theta_i} \\ \boldsymbol{z}_{\theta_i}(t_0) &= 0, \end{aligned} \tag{23}$$

for each $i = 1, \ldots, p$. Thus, the function range test is relatively expensive, and one must consider the tradeoff between the computational cost of the test and the reduction of $\boldsymbol{\Theta}^{(k)}$ that it provides. A mechanism for dealing with this tradeoff is described below in section 5.4.

5.3 Interval-Newton Test

If $\boldsymbol{\Theta}^{(k)}$ has not been eliminated in the objective range test or in the function range test, then the interval-Newton (IN) test is applied. The linear interval equation system

$$\boldsymbol{F}'(\boldsymbol{\Theta}^{(k)})(\boldsymbol{N}^{(k)} - \tilde{\boldsymbol{\theta}}^{(k)}) = -\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^{(k)}), \tag{24}$$

is solved for a new interval $\boldsymbol{N}^{(k)}$, where $\boldsymbol{F}'(\boldsymbol{\Theta}^{(k)})$ is an interval extension of $\boldsymbol{f}'(\boldsymbol{\theta})$, the Jacobian of $\boldsymbol{f}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ (i.e., $\boldsymbol{F}'(\boldsymbol{\Theta}^{(k)})$ is an interval extension of the Hessian of $\phi(\boldsymbol{\theta})$), and $\tilde{\boldsymbol{\theta}}^{(k)}$ is an arbitrary point in $\boldsymbol{\Theta}^{(k)}$. The Taylor model $\boldsymbol{T}_{\boldsymbol{f}'_k}$ of $\boldsymbol{f}'(\boldsymbol{\theta})$ over $\boldsymbol{\Theta}^{(k)}$ is determined and used to bound the coefficients of $\boldsymbol{F}'(\boldsymbol{\Theta}^{(k)})$; that is $\boldsymbol{F}'(\boldsymbol{\Theta}^{(k)}) = B(\boldsymbol{T}_{\boldsymbol{f}'_k})$. It has been shown[6,16,17] that any root of $\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{0}$ (i.e., any stationary point of $\phi(\boldsymbol{\theta})$) contained in $\boldsymbol{\Theta}^{(k)}$ is also contained in the *image* $\boldsymbol{N}^{(k)}$. This implies that if the intersection between $\boldsymbol{\Theta}^{(k)}$ and $\boldsymbol{N}^{(k)}$ is empty, then no root exists in $\boldsymbol{\Theta}^{(k)}$, and also suggests the iterative reduction scheme $\boldsymbol{\Theta}^{(k)} \leftarrow \boldsymbol{\Theta}^{(k)} \cap \boldsymbol{N}^{(k)}$. In addition, it has been shown[6,16,17] that, if $\boldsymbol{N}^{(k)} \subset \boldsymbol{\Theta}^{(k)}$, then there is a *unique* root contained in $\boldsymbol{\Theta}^{(k)}$ and

thus in $\boldsymbol{N}^{(k)}$. Therefore, after computation of $\boldsymbol{N}^{(k)}$ from eq 24, there are three possibilities: 1. $\boldsymbol{\Theta}^{(k)} \cap \boldsymbol{N}^{(k)} = \emptyset$, meaning there is no root in the current interval $\boldsymbol{\Theta}^{(k)}$ and it can be discarded. 2. $\boldsymbol{N}^{(k)} \subset \boldsymbol{\Theta}^{(k)}$, meaning that there is *exactly* one root in the current interval $\boldsymbol{\Theta}^{(k)}$. Thus, the testing of $\boldsymbol{\Theta}^{(k)}$ can stop. The root is enclosed by $\boldsymbol{N}^{(k)} = \boldsymbol{\Theta}^{(k)} \cap \boldsymbol{N}^{(k)}$, and can be more tightly enclosed by repeated application of the IN test, which will converge quadratically to a desired tolerance on the enclosure diameter. 3. Neither of the above, meaning that no conclusion about the number of roots can be drawn, but that $\boldsymbol{\Theta}^{(k)} \leftarrow \boldsymbol{\Theta}^{(k)} \cap \boldsymbol{N}^{(k)}$ can still be applied to try to reduce $\boldsymbol{\Theta}^{(k)}$.

At this point, $\boldsymbol{\Theta}^{(k)}$ has been processed using all three tests (objective range, function range and IN). If the volume of $\boldsymbol{\Theta}^{(k)}$ has been reduced by more than 70%, then the reduced $\boldsymbol{\Theta}^{(k)}$ will be retested, beginning again with the objective range test. Otherwise the reduced $\boldsymbol{\Theta}^{(k)}$ is now bisected, and the resulting two subintervals are added to the sequence (stack) of subintervals to be tested. Various strategies can be used to select the component to be bisected. For the problems solved here, the component with the largest relative width was selected for bisection.

Clearly, the solution of the linear interval equation system given by eq 24 is essential in the IN test. In general, computing the tightest possible interval bounds (interval hull) on the solution of a linear interval equation system is NP-hard,[30] but there are several methods for determining an interval that contains but overestimates the solution. Various IN methods differ in how they solve eq 24 for $\boldsymbol{N}^{(k)}$ and thus in the tightness with which the solution set is enclosed. Frequently, $\boldsymbol{N}^{(k)}$ is computed componentwise using an interval Gauss-Seidel approach, preconditioned with an inverse-midpoint matrix. Gau and Stadtherr[31] proposed a hybrid preconditioning approach (HP/RP). It has been shown to achieve substantially better computational performance than the inverse-midpoint preconditioner alone. Lin and Stadtherr[32,33] have more recently suggested a strategy (LISS_LP) based on linear programming (LP) for solving the linear interval equation system arising in the context of IN methods (eq 24). Exact componentwise bounds on the solution set can be

calculated, while avoiding exponential time complexity. In numerical comparisons[32,33] with HP/RP, LISS_LP has been shown to achieve further improvements in computational performance.

Note that in order to determine a Taylor model of $\boldsymbol{f}'(\boldsymbol{\theta})$, it is necessary to obtain Taylor models for the second-order sensitivities $\boldsymbol{z}_{\theta_j\theta_i} = \partial^2\boldsymbol{z}/\partial\theta_j\partial\theta_i$, $j = 1,\ldots,p$, $i = 1,\ldots,p$, at each t_μ, $\mu = 1,\ldots,r$. To do this, VSPODE is applied to integrate the second-order sensitivity equation,

$$\dot{\boldsymbol{z}}_{\theta_j\theta_i} = \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{z}}\boldsymbol{z}_{\theta_j\theta_i} + \frac{\partial^2\boldsymbol{g}}{\partial\boldsymbol{z}^2}\boldsymbol{z}_{\theta_j}\boldsymbol{z}_{\theta_i} + \frac{\partial^2\boldsymbol{g}}{\partial\boldsymbol{z}\partial\theta_i}\boldsymbol{z}_{\theta_j} + \frac{\partial^2\boldsymbol{g}}{\partial\boldsymbol{z}\partial\theta_j}\boldsymbol{z}_{\theta_i} + \frac{\partial^2\boldsymbol{g}}{\partial\theta_j\partial\theta_i} \tag{25}$$
$$\boldsymbol{z}_{\theta_j\theta_i}(t_0) = 0,$$

for each $j = 1,\ldots,p$ and $i = 1,\ldots,p$. Thus, the IN test is very expensive, and one must consider the tradeoff between its cost and the reduction of $\boldsymbol{\Theta}^{(k)}$ that it provides. A mechanism for dealing with this tradeoff is described in the next section.

5.4 Implementation

The global optimization method consists of three tests, as outlined above. At termination, when all the subintervals in the stack have been tested, all global minimizers of ϕ will have been tightly enclosed. This method can be regarded as a type of branch-and-bound (or branch-and-reduce) scheme on a binary tree. Although the function range and IN tests are effective in reducing and eliminating subdomains from the search space, they are also relatively expensive computationally, especially for optimization in dynamic systems, since these tests require the integration of the sensitivity equations together with the state equations. Thus, in considering the impact on overall computational time, one must consider the tradeoff between the expense of these tests and the reduction of search space that they provide.

Since the IN test and function range test tend to be more effective on smaller intervals, we use a delay scheme, whereby the tests are turned on only by some triggers. The triggers used here are based on the level of the current subinterval $\boldsymbol{\Theta}^{(k)}$ in the binary tree, denoted by $L(\boldsymbol{\Theta}^{(k)})$. That

is, $L(\boldsymbol{\Theta}^{(k)})$ is the number of bisections it has taken to reach $\boldsymbol{\Theta}^{(k)}$. Denote L_F as the trigger level for the function range test, and L_J as the trigger level for the IN test. When $L(\boldsymbol{\Theta}^{(k)}) \geq L_F$, then the function range test will be performed, and when $L(\boldsymbol{\Theta}^{(k)}) \geq L_J$, then the IN test will be performed. In this way, these tests are not applied until the size (volume) of the subinterval $\boldsymbol{\Theta}^{(k)}$ being tested is less than $V_0/2^{L_T}, T \in \{F, J\}$, where V_0 is the size of the initial interval $\boldsymbol{\Theta}^{(0)}$. Since the IN test requires the integration of the second-order sensitivity equations, as well as the first-order conditions, it is always significantly more expensive than the function range test, which requires integration of the first-order sensitivities only. Therefore, when the IN test is triggered, the function range test can always be performed with little additional computational overhead. This means that it would not make sense to set L_F equal to or higher than L_J; thus, in practice, L_F is always less than L_J. We will compare the computational performance of these schemes, including different trigger levels, in the next section.

If the IN test is turned off, then the method becomes a type of finite ϵ-convergence global optimization algorithm. In the objective range test, an ϵ-convergence test is required. That is, if $\hat{\phi} - \underline{B(T_{\Phi_k})} \leq \epsilon^{\text{abs}}$, or $(\hat{\phi} - \underline{B(T_{\Phi_k})})/|\hat{\phi}| \leq \epsilon^{\text{rel}}$, then $\boldsymbol{\Theta}^{(k)}$ will be discarded, where ϵ^{abs} and ϵ^{rel} are absolute and relative convergence tolerances, respectively. In parameter estimation problems, the global minimum in the sum of squares function $\phi(\boldsymbol{\theta})$ can be expected to be close to zero, and there may be also be other local minima with $\phi(\boldsymbol{\theta})$ values close to zero. For this situation, use of an absolute convergence tolerance is inappropriate. For example, if ϵ^{abs} is set to 10^{-3}, then even with the worst case lower bound $\underline{B(T_{\Phi_k})}$, any local minimum with $\hat{\phi} \leq 10^{-3}$ would be accepted as the global minimum, even though a minimum orders of magnitude lower might exist. Thus, for parameter estimation problems, a relative tolerance should be used. Note that, when the IN test is used, the method is an exact global optimization algorithm, i.e., $\epsilon = 0$. In this case, the algorithm will find arbitrarily tight (limited by machine precision) enclosures of all global minimizer points,

with each such enclosure guaranteed to contain a unique stationary point ($\nabla\phi(\boldsymbol{\theta}) = \mathbf{0}$). The width of this enclosure should not be confused with ϵ, which is a tolerance on the objective function value and is zero in this case.

5.5 Summary

A step-by-step summary of the global optimization procedure is given below:

1. Initialization

   (a) Set the trigger levels, L_F and L_J.

   (b) Set the relative convergence tolerance, ϵ^{rel}, or the absolute convergence tolerance, ϵ^{abs}. If the exact global optimum is sought set $\epsilon^{\text{rel}} = 0$ or $\epsilon^{\text{abs}} = 0$.

   (c) Set the current interval $\boldsymbol{\Theta} = \boldsymbol{\Theta}^{(0)}$, and the current level $L_c = 1$.

   (d) Set the interval sequence $\mathcal{L} = \emptyset$.

2. Upper bounding

   (a) Run p^2 local minimizations to initialize $\hat{\phi}$

3. Objective range test

   (a) Compute Taylor models of the states using VSPODE, and then obtain T_ϕ.

   (b) Perform CPP using $T_\phi \leq \hat{\phi}$ to reduce $\boldsymbol{\Theta}$.

   (c) If $\boldsymbol{\Theta} = \boldsymbol{\emptyset}$, go to step 6.

   (d) If $(\hat{\phi} - \underline{B(T_\phi)})/|\hat{\phi}| \leq \epsilon^{\text{rel}}$, or $\hat{\phi} - \underline{B(T_\phi)} \leq \epsilon^{\text{abs}}$, go to step 6.

   (e) If $\overline{B(T_\phi)} < \hat{\phi}$, update $\hat{\phi}$ with local minimization, go to step 3(b).

   (f) If $\boldsymbol{\Theta}$ is sufficiently reduced, go to step 3(a).

4. If $L_c \geq L_J$, perform the function range test and interval-Newton test; else, if $L_c \geq L_F$, perform the function range test. If $\mathbf{\Theta} = \emptyset$, go to step 6. If $\mathbf{\Theta}$ is sufficiently reduced, go to step 3.

5. Branch

   (a) Select the variable on which to bisect (branch) the current interval.

   (b) Bisect the current interval into two subintervals.

   (c) $L_c = L_c + 1$.

   (d) Store one of the two resulting subintervals and L_c in $\mathcal{L}$.

   (e) Set the other subinterval to be the current interval $\mathbf{\Theta}$, and go to step 3.

6. Subinterval selection

   (a) If $\mathcal{L} = \emptyset$, terminate with $\phi^* = \hat{\phi}$.

   (b) Remove one subinterval and associated level information from $\mathcal{L}$ as $\mathbf{\Theta}$ and L_c, and go to step 3.

6 Computational Studies

In this section, results on four example problems are presented. Computational performance with different trigger levels L_F and L_J will be discussed. All example problems were solved on an Intel Pentium 4 3.2GHz machine. The VSPODE package[13] (see section 4), with a $k = 17$ order interval Taylor QR method and a $q = 3$ order Taylor model, was used to integrate the dynamic systems in each problem. Using a smaller value of k will result in the need for smaller step sizes in the integration and so will tend to increase computation time. Using a larger value of q will result in somewhat tighter bounds on the states, though at the expense of additional complexity

in the Taylor model computations. In the ϵ-convergent case, a relative convergence tolerance of $\epsilon^{\text{rel}} = 1 \times 10^{-3}$ was used for all problems. The algorithm was implemented in C++.

6.1 First-order Irreversible Series Reaction

This problem involves parameter estimation for a first-order irreversible chain reaction, as presented by Tjoa and Biegler,[34] and studied by several others.[2,10–12] The reaction system is

$$\text{A} \xrightarrow{k_1} \text{B} \xrightarrow{k_2} \text{C}.$$

Only the concentrations of components A and B are available as measurements. The differential equation model takes the form

$$\begin{aligned} \dot{z}_1 &= -\theta_1 z_1 \\ \dot{z}_2 &= \theta_1 z_1 - \theta_2 z_2 \\ \boldsymbol{z}_0 &= [1,0]^{\text{T}} \qquad t \in [0,1], \end{aligned} \tag{26}$$

where the state vector, $\boldsymbol{z}$, is defined as the concentration vector $[A, B]^{\text{T}}$, and the parameter vector, $\boldsymbol{\theta}$, is defined as $[k_1, k_2]^{\text{T}}$. The measurement data can be found in Esposito and Floudas;[2] it was generated from the model using the parameter values $\boldsymbol{\theta} = [5, 1]^{\text{T}}$, without added error other than roundoff to three significant figures. The initial parameter intervals are $\Theta_1^{(0)} = \Theta_2^{(0)} = [0, 10]$.

Computational results are shown in Table 1. Here, and in subsequent tables of results, L_F and L_J are the trigger levels for the function range test and the IN test, respectively (∞ means the test is turned off); iter is the number of iterations; IFun, IJac, and IN tests are the number of function calls, Jacobian calls, and interval-Newton tests performed, respectively; CPU indicates the CPU time required in seconds. Note that, when $L_J = \infty$, the method becomes an ϵ-convergent algorithm. Otherwise (finite L_J), the method is true global optimization ($\epsilon = 0$). This problem turns out to be very easy. In the ϵ-convergent case ($L_J = \infty$), only 4 iterations and 0.023 seconds

were needed. For exact global optimization, with $L_F = 0$ and $L_J = 1$, only 2 iterations and 0.059 seconds were required. In each case, the method converged to a solution of 1.1858×10^{-6} with the parameter values of $\boldsymbol{\theta} = [5.0035, 1.0000]^{\mathrm{T}}$, which is consistent with the results of Esposito and Floudas.[2]

Comparisons with computation times reported for other methods can give only a very rough idea of the relative efficiency of the methods, due to differences in implementation and in the machine used for the computation. Papamichail and Adjiman[11] reported solving this problem to ϵ-global optimality in 801 seconds using a Matlab implementation on a Sun UltraSPARC-II 360 MHz machine (roughly an order of magnitude slower than the machine used here). Chachuat and Latifi[10] obtained an ϵ-optimal solution to this problem in 280 seconds, using an unspecified machine and a "prototype" implementation. Singer and Barton[12] solved this problem to ϵ-global optimality with an absolute tolerance, so their results are not directly comparable; however, on this problem, the computational cost of their method appears to be similar to the cost of the method given here. All of these other methods provide for ϵ-convergence only.

6.2 First-order Reversible Series Reaction

This example involves a first-order reversible chain reaction, as presented by Tjoa and Biegler[34] and Esposito and Floudas.[2] The reaction system is

$$\mathrm{A} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{B} \underset{k_4}{\overset{k_3}{\rightleftharpoons}} \mathrm{C}.$$

The differential equation model takes the form

$$\begin{aligned} \dot{z}_1 &= -\theta_1 z_1 + \theta_2 z_2 \\ \dot{z}_2 &= \theta_1 z_1 - (\theta_2 + \theta_3) z_2 + \theta_4 z_3 \\ z_3 &= 1 - z_1 - z_2 \\ \boldsymbol{z}_0 &= [1, 0, 0]^{\mathrm{T}} \qquad t \in [0, 1], \end{aligned} \tag{27}$$

where the state vector, $\boldsymbol{z}$, is defined as the concentration vector $[A, B, C]^{\mathrm{T}}$, and the parameter vector, $\boldsymbol{\theta}$, is defined as $[k_1, k_2, k_3, k_4]^{\mathrm{T}}$. In this problem the total number of moles remains constant, so the balance on component C is not independent of the other two component balance equations. Thus, the third model equation is the overall balance. Two sets of measurement data can be found in Esposito and Floudas,[2] both generated from the model using the parameter values $\boldsymbol{\theta} = [4, 2, 40, 20]^{\mathrm{T}}$. One set of data has no added error, and the other set has a small amount of random error added. For the data without added error, the parameter estimation problem is very easy, as also noted by Singer and Barton.[12] Thus, we will concentrate here only on the data set with error added. With this data set, two versions of the parameter estimation problem were solved. The first uses the measurements of the concentration of components of A and B only; the second uses the concentration measurements for all of the components. The initial parameter intervals are $\Theta_1^{(0)} = \Theta_2^{(0)} = [0, 10]$ and $\Theta_3^{(0)} = \Theta_4^{(0)} = [10, 50]$.

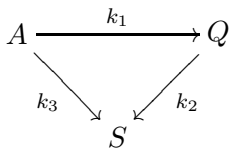
For the first estimation problem (Example 2-1), using only measurements of components A and B, the computational results are shown in Table 2. In all cases, the approach found a global minimum of 8.5727×10^{-4} with the parameter values of $\boldsymbol{\theta} = [4.0186, 2.0451, 40.9670, 20.3336]^{\mathrm{T}}$. For the ϵ-convergent algorithm ($L_J = \infty$), the best performance was 270.8 seconds with $L_F = 15$. Without the function range test ($L_F = \infty$), it took 630.1 seconds to converge. When the IN test is turned on (L_J finite), an exact global optimization algorithm is obtained, and the best performance

was 265.6 seconds with $L_F = 15$ and $L_J = 17$. Thus, the exact algorithm involves no additional cost relative to the ϵ-convergent algorithm. Table 2 (and subsequent tables for other examples) gives results for trigger values on both sides of the best levels. For L_F and L_J values below those shown, the computational cost was significantly higher. On this problem, it appears that, above certain trigger levels, the performance is relatively insensitive to the trigger values used.

For the second estimation problem (Example 2-2), using measurements of all the components, the computational results are shown in Table 3. In all cases, the global minimum found was 1.5875×10^{-3} with the parameter values of $\boldsymbol{\theta} = [4.0202, 2.0517, 39.6473, 19.7246]^{\mathrm{T}}$. This agrees with the solution of Esposito and Floudas,[2] who also solved this version of the problem. For the ϵ-convergent algorithm ($L_J = \infty$), the best performance was 2617.3 seconds without the function range test ($L_F = \infty$). For the exact algorithm (L_J finite), the best performance was 1267.1 seconds with $L_F = 18$ and $L_J = 19$. In this case, the use of the IN test leads to an exact algorithm that is significantly faster than the ϵ-global algorithm. Again, above certain trigger levels, the performance is relatively insensitive to the L_F and L_J values used, though it is more sensitive to L_J than in the previous case.

6.3 Catalytic Cracking of Gas Oil

This problem involves a model representing the catalytic cracking of gas oil (A) to gasoline (Q) and other side products (S), as described by Tjoa and Biegler[34] and also studied by several others.[2,10–12] The reaction is



Only the concentrations of A and Q were measured. Instead of the simple first-order kinetics in the previous two examples, this reaction scheme involves nonlinear reaction kinetics. The differential

equation model takes the form

$$\begin{aligned} \dot{z}_1 &= -(\theta_1 + \theta_3) z_1^2 \\ \dot{z}_2 &= \theta_1 z_1^2 - \theta_2 z_2 \\ \boldsymbol{z}_0 &= [1, 0]^{\mathrm{T}} \qquad t \in [0, 0.95], \end{aligned} \tag{28}$$

where the state vector, $\boldsymbol{z}$, is defined as the concentration vector $[A, Q]^{\mathrm{T}}$ and the parameter vector, $\boldsymbol{\theta}$, is defined as $[k_1, k_2, k_3]^{\mathrm{T}}$. The measurement data was generated using values of the parameters, $\boldsymbol{\theta} = [12, 8, 2]^{\mathrm{T}}$, with a small amount of random error added, and can be found in Esposito and Floudas.[2] The initial parameter intervals are $\Theta_1^{(0)} = \Theta_2^{(0)} = \Theta_3^{(0)} = [0, 20]$.

The computational results are shown in Table 4. In all cases, the global minimum found was 2.6557×10^{-3} with the parameter values of $\boldsymbol{\theta} = [12.2139, 7.9798, 2.2217]^{\mathrm{T}}$, which is consistent with the result of Esposito and Floudas.[2] For the ϵ-global algorithm ($L_J = \infty$), the best performance was 11.1 seconds with $L_F = 11$, while it took 14.3 seconds without the function range test ($L_F = \infty$). With the IN test turned on (exact global algorithm), the best performance was 11.5 seconds with $L_F = 11$ and $L_J = 12$. As in the previous examples, above certain trigger levels, the performance is relatively insensitive to the values used.

Papamichail and Adjiman[11] solved this problem to ϵ-global optimality in 35478 seconds (Sun UltraSPARC-II 360 MHz; Matlab), and Chachuat and Latifi[10] obtained an ϵ-global solution in 10400 seconds (unspecified machine; prototype implementation). Singer and Barton[12] solved this problem to ϵ-global optimality for a series of absolute tolerances, so their results are not directly comparable. However, the computational cost of their method on this problem appears to be similar to the cost of the method given here. These other methods all provide for ϵ-convergence only.

6.4 Lotka-Volterra Predator-Prey Model

This parameter estimation problem, described by Luus,[35] and also studied by Esposito and Floudas,[2] is based on the Lotka-Volterra predator-prey model from theoretical ecology. The system is described by two differential equations,

$$\begin{aligned} \dot{z}_1 &= \theta_1 z_1(1 - z_2) \\ \dot{z}_2 &= \theta_2 z_2(z_1 - 1) \\ \boldsymbol{z}_0 &= [1.2, 1.1]^{\mathrm{T}} \qquad t \in [0, 10], \end{aligned} \tag{29}$$

where z_1 represents the population of the prey and z_2 the population of the predator. The measurement data was generated using the parameter values, $\boldsymbol{\theta} = [3, 1]^{\mathrm{T}}$, with a small amount of normally distributed random error ($\sigma = 0.01$ and zero mean) added to the observations, and can be found in Esposito and Floudas.[2] The initial parameter intervals are $\Theta_1^{(0)} = \Theta_2^{(0)} = [0.1, 10]$.

The computational results are shown in Table 5. In all cases, the global minimum found was 1.2492×10^{-3} with the parameter values of $\boldsymbol{\theta} = [3.2434, 0.9209]^{\mathrm{T}}$. For the ϵ-convergent algorithm ($L_J = \infty$), the best performance is 43.0 seconds without the function range test ($L_F = \infty$). For the exact global algorithm (L_J finite), the best performance is 80.1 seconds with $L_F = 22$ and $L_J = 23$. The sensitivity of performance to trigger levels is a bit higher on this problem, but performance still varies only by a factor of two over a large range of trigger values. The relative insensitivity of the computational performance to the values of L_F and L_J means that this algorithm should be fairly easy to tune, since a wide range of values should give reasonable performance.

7 Concluding Remarks

We have presented here a method for deterministic global optimization in the estimation of parameters for dynamic models (ODE or DAE systems). The method can be implemented as

an ϵ-global algorithm, or, by use of the interval-Newton method, as an exact algorithm. In the latter case, the method provides a mathematically guaranteed and computationally validated global optimum in the goodness of fit function. On some problems, the exact global optimization algorithm requires some additional computational effort relative to the ϵ-convergent case. However, on other problems, the exact algorithm is actually less costly. Other global optimization algorithms proposed for solving this problem either offer no theoretical guarantees,[2] or provide ϵ-convergence only.[10–12] The exact algorithm described here provides a rigorous and validated global optimum, and does so with a computational cost that is comparable to or better than the other methods.

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.

List of Symbols

B = interval bound

f, g, h = functions

F = interval extension of function f

k = order of ITS method

J = set of state variables whose derivatives appear in the constraints

L = level of subinterval in the binary tree

$\mathcal{L}$ = the interval sequence

M = set of fitted state variables

p = polynomial part of Taylor model; number of parameters

q = order of Taylor model

r = number of data points

R = remainder part of Taylor model

t = independent variable in ODEs

T = Taylor model

x = real variable

X = interval variable

y = state variable (real) in ODEs (section 4)

Y = state variable (interval) in ODEs (section 4)

z = state variable in ODE constraints

$\bar{z}$ = measured value of z

Z = set of state variables

Greek symbols

ϵ = convergence tolerance on objective value

ϕ = objective function (real)

$\hat{\phi}$ = upper bound on global minimum of ϕ

ϕ^* = global minimum of ϕ

Φ = objective function (interval)

θ = parameter (real)

Θ = parameter (interval)

Superscripts

abs = absolute

rel = relative

$[i]$ = refers to i-th Taylor coefficient in Taylor expansion with respect to time

(k) = refers to k-th subinterval tested

Subscripts

0 = initial value; base point in Taylor expansion (section 3.2)

c = constraint (section 3.3); current subinterval (section 5.5)

References

(1) Bard, Y. *Nonlinear Parameter Estimation;* Academic Press: New York, 1974.

(2) Esposito, W. R.; Floudas, C. A. Global optimization for the parameter estimation of differential-algebraic systems. *Ind. Eng. Chem. Res.* **2000,** *39,* 1291-1310.

(3) Stewart, W. E.; Caracotsios, M.; Sorensen, J. P. Parameter estimation from multiresponse data. *AIChE J.* **1992,** *38,* 641-650.

(4) Boender, C. E.; Romeijn, H. E. Stochastic Methods. In *Handbook of Global Optimization*; Horst, R.; Pardalos, P. M., Eds.; Kluwer Academic Publishers: Dordrecht, 1995.

(5) Floudas, C. A. *Deterministic Global Optimization: Theory, Methods and Application;* Kluwer Academic Publishers: Dordrecht, The Netherlands, 2000.

(6) Hansen, E.; Walster, G. W. *Global Optimization Using Interval Analysis;* Marcel Dekker: New York, 2004.

(7) Adjiman, C. S.; Androulakis, I. P.; Floudas, C. A.; Neumaier, A. A global optimization method, αBB, for general twice-differentiable NLPs–I. Theoretical advances. *Comput. Chem. Eng.* **1998,** *22,* 1137-1158.

(8) Adjiman, C. S.; Dallwig, S.; Floudas, C. A.; Neumaier, A. A global optimization method, αBB, for general twice-differentiable NLPs–II. Implementation and computational results. *Comput. Chem. Eng.* **1998,** *22,* 1159-1179.

(9) Papamichail, I.; Adjiman, C. S. A rigorous global optimization algorithm for problems with ordinary differential equations. *J. Global. Opt.* **2002,** *24,* 1-33.

(10) Chachuat, B.; Latifi, M. A. A new approach in deteterministic global optimisation of problems with ordinary differential equations. In *Frontiers in Global Optimization*; Floudas, C. A.; Pardalos, P. M., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2004.

(11) Papamichail, I.; Adjiman, C. S. Global optimization of dynamic systems. *Comput. Chem. Eng.* **2004,** *28,* 403-415.

(12) Singer, A. B.; Barton, P. I. Global optimization with nonlinear ordinary differential equations. *J. Global Optim.* **2006,** *34,* 159-190.

(13) Lin, Y.; Stadtherr, M. A. Validated solutions of initial value problems for parametric ODEs. *SIAM J. Sci. Comput.* **2005,** submitted.

(14) Brenan, K. E.; Campbell, S. L.; Petzold, L. R. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations;* SIAM: Philadelphia, 1996.

(15) Jaulin, L.; Kieffer, M.; Didrit, O.; É Walter, *Applied Interval Analysis;* Springer-Verlag: London, 2001.

(16) Kearfott, R. B. *Rigorous Global Search: Continuous Problems;* Kluwer Academic Publishers: Dordrecht, The Netherlands, 1996.

(17) Neumaier, A. *Interval Methods for Systems of Equations;* Cambridge University Press: Cambridge, UK, 1990.

(18) Makino, K.; Berz, M. Remainder differential algebras and their applications. In *Computational Differentiation: Techniques, Applications, and Tools*; Berz, M.; Bishof, C.; Corliss, G.; Griewank, A., Eds.; SIAM: Philadelphia, 1996.

(19) Makino, K.; Berz, M. Efficient control of the dependency problem based on Taylor model methods. *Reliab. Comput.* **1999,** *5,* 3-12.

(20) Makino, K.; Berz, M. Taylor models and other validated functional inclusion methods. *Int. J. Pure Appl. Math.* **2003,** *4,* 379-456.

(21) Revol, N.; Makino, K.; Berz, M. Taylor models and floating-point arithmetic: Proof that arithmetic operations are validated in COSY. *J. Logic Algebr. Progr.* **2005,** *64,* 135-154.

(22) Makino, K. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators,* Thesis, Michigan State University, East Lansing, Michigan, USA, 1998.

(23) Neumaier, A. Taylor forms - Use and limits. *Reliab. Comput.* **2002,** *9,* 43-79.

(24) Moore, R. E. *Interval Analysis;* Prentice-Hall: Englewood Cliffs, NJ, 1966.

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

(26) Lohner, R. J. Computations of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In *Computational Ordinary Differential Equations*; Cash, J.; Gladwell, I., Eds.; Clarendon Press: Oxford, UK, 1992.

(27) Nedialkov, N. S.; Jackson, K. R.; Pryce, J. D. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliab. Comput.* **2001,** *7,* 449-465.

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

(29) Byrd, R. H.; Lu, P.; Nocedal, J.; Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci Comput.* **1995,** *16,* 1190-1208.

(30) Rohn, J.; Kreinovich, V. Computing exact componentwise bounds on solution of linear systems with interval data is NP-hard. *SIAM J. Matrix Anal. Appl.* **1995,** *16,* 415-420.

(31) Gau, C.-Y.; Stadtherr, M. A. New interval methodologies for reliable chemical process modeling. *Comput. Chem. Eng.* **2002,** *26,* 827-840.

(32) Lin, Y.; Stadtherr, M. A. Advances in interval methods for deterministic global optimization in chemical engineering. *J. Global Optim.* **2004,** *29,* 281-296.

(33) Lin, Y.; Stadtherr, M. A. LP strategy for the interval-Newton method in deterministic global optimization. *Ind. Eng. Chem. Res.* **2004,** *43,* 3741-3749.

(34) Tjoa, T. B.; Biegler, L. T. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Ind. Eng. Chem. Res.* **1991,** *30,* 376.

(35) Luus, R. Parameter estimation of Lotka-Volterra problem by direct search optimization. *Hung. J. Ind. Chem.* **1998,** *26,* 287.

Algorithm 1 VSPODE_Phase2(In: $\hat{\boldsymbol{T}}_{\boldsymbol{y}_j}$, A_j, $\boldsymbol{V}_j$, h_j, $\tilde{\boldsymbol{Y}}_j$, $\boldsymbol{Y}_j$; Out: $\boldsymbol{T}_{\boldsymbol{y}_{j+1}}$, $\hat{\boldsymbol{T}}_{\boldsymbol{y}_{j+1}}$, A_{j+1}, $\boldsymbol{V}_{j+1}$)[a]

1: $\boldsymbol{Z}_{j+1} = h_j^k \boldsymbol{F}^{[k]}(\tilde{\boldsymbol{Y}}_j, \boldsymbol{\Theta})$

2: $\boldsymbol{T}_{\boldsymbol{U}_{j+1}} = \hat{\boldsymbol{T}}_{\boldsymbol{y}_j} + \sum_{i=1}^{k-1} h_j^i \boldsymbol{T}_{\hat{\boldsymbol{f}}^{[i]}} + \boldsymbol{Z}_{j+1}$

3: $\boldsymbol{S}_j = I + \sum_{i=1}^{k-1} h_j^i \boldsymbol{J}(\boldsymbol{f}^{[i]}; \boldsymbol{Y}_j, \boldsymbol{\Theta})$

4: $A_{j+1} = (m(\boldsymbol{S}_j A_j))^{-1}$

5: $(\hat{\boldsymbol{T}}_{\boldsymbol{y}_{j+1}}, \boldsymbol{R}_{\boldsymbol{U}_{j+1}}) \Leftarrow \boldsymbol{T}_{\boldsymbol{U}_{j+1}}$, with $m(\boldsymbol{R}_{\boldsymbol{U}_{j+1}}) = \boldsymbol{0}$

6: $\boldsymbol{V}_{j+1} = (A_{j+1}^{-1} \boldsymbol{S}_j A_j) \boldsymbol{V}_j + A_{j+1}^{-1} \boldsymbol{R}_{\boldsymbol{U}_{j+1}}$

7: $\boldsymbol{T}_{\boldsymbol{y}_{j+1}} = \hat{\boldsymbol{T}}_{\boldsymbol{y}_{j+1}} + A_{j+1} \boldsymbol{V}_{j+1}$

[a]The procedure begins with $\boldsymbol{V}_0 = \boldsymbol{0}$, $\hat{\boldsymbol{T}}_{\boldsymbol{y}_0} = (\boldsymbol{y}_0, [0, 0])$, and $A_0 = I$. $\boldsymbol{J}(\boldsymbol{f}^{[i]}; \boldsymbol{Y}_j, \boldsymbol{\Theta})$ denotes the interval extension of the Jacobian (with respect to the state variables) of $\boldsymbol{f}^{[i]}$ over $\boldsymbol{y}_j \in \boldsymbol{Y}_j$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, and $\boldsymbol{T}_{\hat{\boldsymbol{f}}^{[i]}} = \boldsymbol{f}^{[i]}(\hat{\boldsymbol{T}}_{\boldsymbol{y}_j}, \boldsymbol{T}_{\boldsymbol{\theta}})$.

Table 1: Results[a] for Example 1

Algorithm	L_F	L_J	iter	IFun	IJac	IN tests	CPU s
ϵ-Global	∞	∞	4	0	0	0	0.023
Exact	0	1	2	1	1	1	0.059

[a] L_F = trigger level for function range test; L_J = trigger level for interval-Newton test; iter = number of iteration; IFun = number of function calls; IJac = number of Jacobian calls; IN tests = number of interval-Newton tests; CPU = CPU time (s)

Table 2: Results[a] for Example 2-1

Algorithm	L_F	L_J	iter	IFun	IJac	IN tests	CPU s
ϵ-Global	14	∞	1478	477	0	0	296.4
	15	∞	1622	395	0	0	270.8
	16	∞	1886	427	0	0	311.5
	17	∞	2162	405	0	0	320.1
	18	∞	2485	394	0	0	333.7
	19	∞	2862	433	0	0	376.0
	∞	∞	9050	0	0	0	630.1
Exact	14	15	1357	146	235	194	811.7
	15	16	1380	159	48	42	297.8
	15	17	1392	188	30	29	265.6
	16	17	1641	195	39	36	312.6
	16	18	1671	226	36	32	315.1
	17	18	1952	214	35	26	337.6
	17	19	1989	247	32	22	336.6
	18	19	2301	239	26	18	347.8

[a] L_F = trigger level for function range test; L_J = trigger level for interval-Newton test; iter = number of iteration; IFun = number of function calls; IJac = number of Jacobian calls; IN tests = number of interval-Newton tests; CPU = CPU time (s)

Table 3: Results[a] for Example 2-2

Algorithm	L_F	L_J	iter	IFun	IJac	IN tests	CPU s
ϵ-Global	16	∞	10192	5873	0	0	2978.1
	17	∞	10645	5760	0	0	2888.0
	18	∞	11296	5693	0	0	2883.2
	19	∞	12255	5561	0	0	2897.4
	20	∞	13223	5386	0	0	2915.5
	21	∞	14391	5290	0	0	2971.6
	22	∞	15826	5102	0	0	2968.5
	∞	∞	40552	0	0	0	2617.3
Exact	17	19	3775	791	315	257	1430.2
	18	19	4330	694	240	220	1267.1
	18	20	4737	928	335	272	1613.6
	18	21	5173	1257	343	305	1813.2
	19	20	5576	766	272	256	1437.9
	19	21	6072	1038	364	292	1870.1
	19	22	6491	1394	318	288	1901.8
	20	21	6925	854	294	262	1655.6
	20	22	7401	1145	372	315	2004.2
	21	22	8490	1052	302	289	1912.6

[a] L_F = trigger level for function range test; L_J = trigger level for interval-Newton test; iter = number of iteration; IFun = number of function calls; IJac = number of Jacobian calls; IN tests = number of interval-Newton tests; CPU = CPU time (s)

Table 4: Results[a] for Example 3

Algorithm	L_F	L_J	iter	IFun	IJac	IN tests	CPU s
ϵ-Global	10	∞	150	28	0	0	11.8
	11	∞	182	19	0	0	11.1
	12	∞	206	15	0	0	11.3
	13	∞	226	14	0	0	11.9
	14	∞	247	13	0	0	12.5
	15	∞	266	13	0	0	13.3
	16	∞	285	12	0	0	13.9
	∞	∞	359	0	0	0	14.3
Exact	10	11	149	25	3	1	13.1
	11	12	181	18	1	1	11.5
	12	13	206	14	1	1	11.7
	13	14	226	13	1	1	12.3
	14	15	247	13	1	1	13.0

[a] L_F = trigger level for function range test; L_J = trigger level for interval-Newton test; iter = number of iteration; IFun = number of function calls; IJac = number of Jacobian calls; IN tests = number of interval-Newton tests; CPU = CPU time (s)

Table 5: Results[a] for Example 4

Algorithm	L_F	L_J	iter	IFun	IJac	IN tests	CPU s
ϵ-Global	17	∞	501	87	0	0	104.0
	18	∞	502	69	0	0	89.3
	19	∞	494	54	0	0	78.7
	20	∞	501	48	0	0	74.9
	∞	∞	536	0	0	0	43.0
Exact	17	18	473	14	56	17	235.0
	18	19	473	9	45	21	201.3
	19	20	466	11	31	22	154.0
	20	21	471	14	21	17	122.3
	21	22	481	14	13	13	93.4
	22	23	498	8	10	10	80.1
	23	24	514	12	12	11	94.7
	24	25	531	15	12	12	97.7

[a] L_F = trigger level for function range test; L_J = trigger level for interval-Newton test; iter = number of iteration; IFun = number of function calls; IJac = number of Jacobian calls; IN tests = number of interval-Newton tests; CPU = CPU time (s)