Noname manuscript No.
(will be inserted by the editor)

Error bounds for initial value problems by optimization

Qaisra Fazal · **Arnold Neumaier**

Received: date / Accepted: date

Abstract We use preconditioned defect estimates and optimization techniques to compute error bounds for approximate solutions of initial value problems for ordinary differential equations with uncertain initial conditions.

The bounds are based on new results about conditional differential inequalities, which are developed, too.

The scheme is implemented in MATLAB and AMPL, and the resulting enclosures are compared with the packages VALENCIA-IVP, VNODE-LP and VSPODE for bounding solutions of ODEs. The current prototype implementation uses heuristics to solve the global optimization subproblems, hence the bounds obtained in the numerical experiments are not fully rigorous. They could be made so by using rigorous global optimization and rounding error control, but the effect on the bounds is likely to be marginal only.

Keywords Validated enclosures · IVP · conditional differential inequality · ellipsoid enclosure · global optimization.

Mathematics Subject Classification (2000)

Arnold Neumaier
University of Vienna
Faculty of Mathematics
Nordbergstr. 15, 1090 Wien, Austria
Tel. +43-1-427750661
E-mail: Arnold.Neumaier@univie.ac.at

Qaisra Fazal
University of Vienna, Faculty of Mathematics
Nordbergstr. 15, 1090 Wien, Austria
Tel. +43-1-427750705
E-mail: qaisra.fazal@univie.ac.at

1 Introduction

1.1 Problem formulation

The problem addressed in this paper is the computation of error bounds for a system of n ordinary differential equations (ODEs) with uncertain initial data ranging over an n-dimensional box,

$$y'(t) = F(t,y(t)),\ \ y(0) \in \boldsymbol{y}_0,\ t \in [0,T], \tag{1}$$

where $F \in C^1(D,\mathbb{R}^n)$ is a continuously differentiable function from $D \subseteq \mathbb{R} \times \mathbb{R}^n$ to $\mathbb{R}^n$, $y : [0,T] \to \mathbb{R}^n$ is the unknown function to be solved for, and $\boldsymbol{y}_0 = [\underline{y}_0, \overline{y}_0]$ is a box, with componentwise bounds. The bounds computed for $y(t)$ are of the ellipsoidal form

$$\|U(t)(y(t) - u(t))\| \le \overline{\Delta}(t), \tag{2}$$

where $u(t)$ is an approximate solution of the systems of ODEs, $U(t)$ is a matrix function used to precondition the system, and $\overline{\Delta}(t)$ is a bound for the preconditioned error. The preconditioner is introduced to reduce the wrapping effect caused by the outer ellipsoidal approximation.

1.2 Prior work

I. Theory: Bounding the solution of ordinary differential equations in the context of interval analysis was first discussed by MOORE [31,33]. He designed a computer program to determine intervals containing the exact solution to an ordinary differential equation [34,45]. Then in [32], he presented a technique to reduce the wrapping effect produced by the successive expansions into Taylor series with interval remainder [31,45].

EIJGENRAAM [11] developed an algorithm for the solution of initial value problems (IVPs) with initial values contained in a given initial value set. He gave the first rigorous overestimation analysis for an enclosure algorithm.

CHERNOUSKO [8,5,7,6] developed methods for ellipsoidal enclosures in place of interval arithmetic, but did not account for discretization errors. KURZHANSKI [18,20,19] and FILIPPOVA [13] also used ellipsoidal techniques for reachability analysis.

KÜHN [17], derived error bounds for a computed approximate solution, based on defect estimates and bounds on the second derivatives of F. Kühn's approach is quite different from that of Moore, and is superior for highly nonlinear problems.

NEDIALKOV [38] presented an interval Hermite-Obreschkoff (IHO) method for computing guaranteed enclosures of the solution of IVPs. He gave a comparison between interval Taylor series (ITS) methods and his IHO scheme for the same stepsize and order, and showed that IHO scheme has a smaller truncation error, better stability, and requires fewer Taylor coefficients and high-order Jacobians.

II. Software packages There are several software packages for computing error bounds for initial value problems for systems of ODEs. These include AWA [25,26,27], VNODE-LP [36,35,37,38], VSPODE [23,24], VODESIA [9], ADIODES [48], COSY INFINITY [4,28,29,30] and RIOT [10]. They are based on the traditional approach, using interval arithmetic and a Taylor expansion in time (and in some cases also in uncertainty space), refining the earliest enclosure method by MOORE [33]. Each step proceeds according to the following scheme, but the details are different in each implementation and characterize a particular method.

I: (Existence and uniqueness) Find a step size h_i and a coarse enclosure box $\boldsymbol{y_i} \subseteq D$ such that for $t \in \boldsymbol{t_i} := [t_i, t_i + h_i]$, the solution $y(t)$ exists and satisfies $y(t) \in \boldsymbol{y_i}$.

II: (Tightening) Compute a tight enclosure for $y(t)$ at $t = t_{i+1} := t_i + h_i$.

Historically, AWA, written in Pascal-XSC, was one of the most important packages for computing guaranteed enclosures for the solution of ordinary IVP involving parameters with uncertainties. It replaced Moore's simple preconditioner by a preconditioner based on *QR* factorizations, thereby increasing the numerical stability of Moore's method.

VNODE-LP (a successor of the VNODE package) is a C++ package developed using literate programming. It is based on an IHO method for performing the tightening step II.

VSPODE is a C++ package that allows uncertainty in parameters and initial data; these are dealt with by using Taylor models.

VALENCIA-IVP [44,42,43], written in C++, first calculates a non-validated approximate solution of an initial value problem. Based on this, a fixed-point iterative scheme is implemented to compute guaranteed error bounds.

ADIODES [48] is a C++ package based on the interval package BIAS/PROFIL[16]. It finds periodic solutions of ordinary differential equations by using Picard iteration and uses the extended mean value method to compute the enclosures.

The automatic differentiation(AD) packages FADBAD, TADIFF[3] and FADBAD++ [2] are used in VNODE, VNODE-LP, VSPODE, and ADIODES to compute the Taylor coefficients of the solution of ODE and the solution of its variational equation, while in VALENCIA-IVP, Jacobians must be programmed explicitly.

COSY INFINITY is a FORTRAN based code to study the design of beam physics system. This provides a direct functional relationship between final and initial solution sets by means of Taylor model. It also computes tight enclosures by constructing Taylor models of the flow of ODE.

RIOT, written in C++, is a verified Taylor-model-based IVP solver.

1.3 Contents of this paper

I. Theory: The mathematical techniques proposed in this paper are fully rigorous and are refinement of the theoretical approaches of CHERNOUSKO and KÜHN [17]. Our defect estimates are based on new differential inequalities (which replace the second derivative estimates of Kühn), applied together with global optimization of the constants involved in the theoretical bounds. The bounds are applied to the squared Euclidean norm of a preconditioned residual and define ellipsoidal bounds for the solutions. The initial ellipsoidal enclosure must contain the given initial box; this results in an initial wrapping step not present in the traditional methods (where an initial wrapping step would instead be present if the initial uncertainty has an ellipsoidal form). Nevertheless, the improved bounding procedures for the ellipsoidal information make up for this wrapping except in the simplest problems.

The new conditional differential inequalities are derived in Section 2, and used in Section 3 to derive theoretical error bounds. These form the backbone of the new method presented in Section 4.

II. Implementation: To test the new method, we provided a prototype implementation in MATLAB in the solver DIBIS (Differential Inequalities Based IVP Solver). A rigorous implementation would require a rigorous constraint global optimization solver, but there is no such solver in MATLAB. Thus we used a simple (theoretically unreliable) multistart heuristic for global optimization, using many starting points to make it work correctly in our tests.

The implementation would become fully rigorous with an interface to a rigorous global optimization solver such as COCONUT [47]; it is planned to do this in the near future.

In Section 5, numerical results of the new method are shown. The resulting enclosures of ODEs are compared with those of the existing solvers VALENCIA-IVP, VNODE-LP and VSPODE.

In Section 6, we briefly describe perspectives and future work.

2 Conditional differential inequalities

An inequality such as $y'(t) \geq F(t, y(t))$ connecting a function, its arguments, and its derivative is known as a differential inequality.

The theory of differential inequalities plays an essential role for our enclosures of ODEs. It dates back to a 1919 paper by GRONWALL [14]; a systematic exposition was given in 1964 by WALTER [50]. In 1969, LAKSHMIKANTHAM & LEELA [22] used differential inequalities to construct sup and sub functions for differential equations. In 1980, ADAMS [1] used differential inequalities to characterize the systems of ODEs for which tight inclusions can be computed. In 1983, NICKEL [21] used differential inequalities to compute bounds for the set of solutions of functional-differential equations. In 1994, NEUMAIER [39] used logarithmic norms and differential inequalities to compute rigorous enclosures for the solution of dissipative differential equations over very long times, and outlined an idea for obtaining enclosures without step size restrictions for the solutions of certain stiff ODEs. In 2002, PAPAMICHAIL & ADJIMAN [40] used differential inequalities to construct bounds on the set of solutions of parameter-dependent ODEs and their second order sensitivities.

Our new contribution to the theory is the concept of a conditional differential inequality. This leads to flexible and powerful bounds, which are more suitable for adaptive usage than traditional differential inequalities. Conditional differential inequalities form the basis of our new method for computing bounds on solutions of systems of ODEs.

Our conditional differential inequalities come in two versions, nearly symmetric to each other.

Theorem 1 *Let $\Delta : [\underline{t}, \bar{t}] \to \mathbf{R}$ be a continuously differentiable function and let Δ_0, Δ_+ and Δ'_+ be constants such that, for all $t \in [\underline{t}, \bar{t}]$,*

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) \leq \Delta'_+ \tag{3}$$

If $\Delta(\underline{t}) \leq \Delta_0 < \Delta_+$ and $\Delta'_+ > 0$ then

$$\Delta(t) \leq \Delta_0 + \Delta_1 (t - \underline{t}) \quad \text{for } t \in [\underline{t}, t_1], \tag{4}$$

where the constants t_1 and Δ_1 are defined by

$$t_1 := \bar{t}, \quad \Delta_1 := 0 \quad \text{if } \Delta'_+ \leq 0, \tag{5}$$

$$\left.\begin{array}{l} t_1 := \min\left(\bar{t}, \underline{t} + \frac{\Delta_+ - \Delta_0}{\Delta'_+}\right) \\ \Delta_1 := \Delta'_+ \end{array}\right\} \quad \text{if } \Delta'_+ > 0. \tag{6}$$

Proof We first make the stronger assumption that

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) < \Delta'_+, \;\; \text{for all } t \in [\underline{t}, \bar{t}] \tag{7}$$

and show that (4) holds with (5) and (6).

Let t_0 be the supremum of all $t_1 \leq \bar{t}$ such that (4) holds. We define $\delta(t) := \Delta_0 + \Delta_1 (t - \underline{t}) - \Delta(t)$ and have $\delta(\underline{t}) = \Delta_0 - \Delta(\underline{t}) \geq 0$. If $\delta(\underline{t}) > 0$ then

$$\delta(t') = \delta(\underline{t}) + \delta'(\underline{t})(t' - \underline{t}) + o(t' - \underline{t}) > 0 \text{ for small } t' > \underline{t}. \tag{8}$$

Otherwise $\Delta(\underline{t}) = \Delta_0$, hence $\delta'(\underline{t}) = \Delta_1 - \Delta'(\underline{t}) \geq \Delta'_+ - \Delta'(\underline{t}) > 0$ by (7), and we see that (8) holds too. Thus (8) holds generally.

By continuity, we see that (4) also holds for t_0 in place of t_1. Therefore, if (4) is violated, then $t_0 < t_1$, and for some sequence $t_l \downarrow t_0$, we have

$$\Delta(t_l) > \Delta_0 + \Delta_1 (t_l - \underline{t}).$$

Taking the limit, we find that

$$\Delta(t_0) = \Delta_0 + \Delta_1 (t_0 - \underline{t}) \geq \Delta_0, \tag{9}$$

$$\Delta'(t_0) = \lim_{t_l \to t_0} \frac{\Delta(t_l) - \Delta(t_0)}{t_l - t_0} \geq \Delta_1 \geq \Delta'_+. \tag{10}$$

Because of (7), (9) and (10), we see that $\Delta(t_0) > \Delta_+$. But then (9) implies $\Delta_1 (t_0 - \underline{t}) > \Delta_+ - \Delta_0 > 0$, hence

$$\Delta_1 > 0, \; t_0 > \underline{t} + \frac{\Delta_+ - \Delta_0}{\Delta_1} \geq t_1, \tag{11}$$

a contradiction.

Now we return to the general case and assume that (3) holds. Then for all $\varepsilon > 0$, we see that

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) < \Delta'_+ + \varepsilon. \tag{12}$$

Therefore, (4) holds with

$$t_1 = \bar{t}, \quad \Delta_1 = 0, \quad \text{if } \Delta'_+ + \varepsilon \leq 0 \tag{13}$$

and

$$t_1 = \min\left(\bar{t}, \underline{t} + \frac{\Delta_+ - \Delta_0}{\Delta'_+ + \varepsilon}\right), \quad \Delta_1 = \Delta'_+ + \varepsilon, \quad \text{if } \Delta'_+ + \varepsilon > 0. \tag{14}$$

We now distinguish three cases from (14) as follows:

(i) If $\Delta'_+ > 0$ then, for $\varepsilon \to 0$, we have

$$t_1 = \underline{t} + \frac{\Delta_+ - \Delta_0}{\Delta'_+}, \quad \Delta_1 = \Delta'_+ > 0,$$

so (4) holds with (6).

(ii) If $\Delta'_+ = 0$ then $t_1 = \bar{t}$, $\Delta_1 = 0$, so (4) holds with (5).
(iii) If $\Delta'_+ < 0$ then, for small $\varepsilon > 0$, we have (13), so for $\varepsilon \to 0$, (4) holds with (5). Thus Theorem 1 holds in each case.

Theorem 2 *Let $\Delta : [\underline{t},\bar{t}] \to \mathbf{R}$ be a continuously differentiable function and let Δ_0, Δ_- and Δ'_- be constants such that, for all $t \in [\underline{t},\bar{t}]$,*

$$\Delta(t) \in [\Delta_-,\Delta_0] \;\Rightarrow\; \Delta'(t) \le \Delta'_-. \tag{15}$$

If $\Delta_- < \Delta_0$ and $\Delta'_- < 0$ then

$$\Delta(t) \le \Delta_0 + \Delta_1(t-\underline{t}) \quad \text{for } t \in [\underline{t},t_1], \tag{16}$$

where the constants t_1 and Δ_1 are defined by

$$t_1 := \min\Big(\bar{t}, \underline{t} + \frac{\Delta_- - \Delta_0}{\Delta'_-}\Big), \quad \Delta_1 := \Delta'_-. \tag{17}$$

Proof This is proved by essentially the same argument as in the proof of Theorem 1.

In our algorithm, Theorem 1 is applied when bounds growth is expected (see Case I in Section 3), while Theorem 2 is used when the bounds are expected to decay (see Case II in Section 3).

3 Error bounds by optimization

In this section, we discuss our new scheme for computing validated enclosures of the solutions of ODEs. The techniques presented by CHERNOUSKO [5] and KÜHN [17] are combined with the new bounds from Theorem 1 and Theorem 2, evaluated with global optimization methods to compute error bounds. We begin with computing an approximate solution of the differential equations and a t-dependent preconditioning matrix defining the shape of the ellipsoid. We then compute the worst case of the norm of the preconditioned defect by solving a global optimization problem, which produces a value for the size of the enclosing ellipsoid.

The initial enclosure. The initial box $\boldsymbol{y}_0$ is contained in the ellipsoid

$$\mathscr{E} = \{y \in \mathbb{R}^n \mid \|U_0(y-u_0)\|^2 \le \Delta_0\}, \tag{18}$$

where $u_0 = \frac{1}{2}(\bar{y}_0 + \underline{y}_0)$ and $\Delta_0 = \|U_0(\boldsymbol{y_0} - u_0)\|^2$. We choose as initial preconditioner $U_0 := \mathrm{Diag}\,(\bar{y}_0 - \underline{y}_0)^{-1}$, so that $\Delta_0 = \frac{n}{4}$.

Approximating the solution. We use a standard solver to get, on an adaptive grid, at time t_i an approximate solution $u(t_i)$ of the IVP posed at the midpoint of the initial box, and an approximate solution $U(t_i)$ of the variational problem for the preconditioner with initial condition $U(0) = U_0$, modified by a nonlinear regularization term to control the condition of the preconditioner:

$$Y(t)' = -YF_y - Y\Big(\delta_0 + \delta_k (Y^T Y)^k\Big), \quad t \in [0,T]. \tag{19}$$

Here δ_0 and δ_k are positive regularization parameters, and k is a nonnegative integer. In our current implementation, these parameters are chosen interactively for each problem.

The solutions on the grid are interpolated by cubic Hermite splines to get continuous approximations $u(t)$ of the solution and $U(t)$ of the preconditioner.

Given $\underline{t} < \bar{t}$, we define piecewise continuously differentiable solutions u and U by using cubic Hermite interpolation:

$$u(t) = a(t-\underline{t})^2(t-\bar{t}) + b(t-\underline{t})^2 + c(t-\underline{t}) + d, \;\; t \in [\underline{t},\bar{t}], \tag{20}$$

$$u(t)' = 2a(t-\underline{t})(t-\bar{t}) + a(t-\underline{t})^2 + 2b(t-\underline{t}) + c, \;\; t \in [\underline{t},\bar{t}], \tag{21}$$

and

$$U(t) = A(t-\underline{t})^2(t-\bar{t}) + B(t-\underline{t})^2 + C(t-\underline{t}) + D, \;\; t \in [\underline{t},\bar{t}], \tag{22}$$

$$U(t)' = 2A(t-\underline{t})(t-\bar{t}) + A(t-\underline{t})^2 + 2B(t-\underline{t}) + C, \;\; t \in [\underline{t},\bar{t}]. \tag{23}$$

Here the coefficients a,b,c,d and A,B,C and D are computed by using the continuity of the function and its first derivative and are given as:

$$\begin{aligned}
a &:= \Big(u(\bar{t})' + u(\underline{t})' - 2(u(\bar{t}) - u(\underline{t}))/h\Big)/h^2,\\
b &:= \Big((u(\bar{t}) - u(\underline{t}))/h - u(\underline{t})'\Big)/h,\\
c &:= u(\underline{t})',\\
d &:= u(\underline{t}),
\end{aligned} \tag{24}$$

and similarly

$$\begin{aligned}
A &:= \Big(U(\bar{t})' + U(\underline{t})' - 2(U(\bar{t}) - U(\underline{t}))/h\Big)/h^2,\\
B &:= \Big((U(\bar{t}) - U(\underline{t}))/h - U(\underline{t})'\Big)/h,\\
C &:= U(\underline{t})',\\
D &:= U(\underline{t}).
\end{aligned} \tag{25}$$

Propagating error bounds. We want to find error bounds for an approximate continuously differentiable solution $u(t) \approx y(t)$ of (1). The error between the approximate and the exact solution is given by

$$\varepsilon(t) = y(t) - u(t).$$

We define the preconditioned residual

$$\eta(t) = U(t)\varepsilon(t), \tag{26}$$

and look for bounds of

$$\Delta(t) = \|\eta(t)\|^2 = \eta(t)^T\eta(t) = \sum_{i=1}^{n} \eta_i^2(t) \ge 0, \tag{27}$$

where $\|.\|$ denotes the Euclidean norm. Bounds at $t = 0$ come from the initial enclosure. Thus we may assume that we have already such a bound for $t = \underline{t}$,

$$0 \le \Delta(\underline{t}) \le \Delta_0. \tag{28}$$

We want to extend it to a bound valid for t in some interval $[\underline{t},\bar{t}]$. To find these bounds, we differentiate (27),

$$\begin{aligned}\Delta' &= 2\eta^T\eta' = 2\eta^T(U\varepsilon)'\\ &= 2(U\varepsilon)^T(U\varepsilon'+U'\varepsilon)\\ &= 2(U\varepsilon)^T\Big(U(F(t,u+\varepsilon)-u')+U'\varepsilon\Big).\end{aligned}$$

This is done by solving two auxiliary global optimization problems, accounting for two qualitatively different situations. The actual step size is determined by the maximal time for which the above theorems are applicable.

CASE I: *Bound growth expected.* Denote by Δ_1' the maximum value of the optimization problem

$$\begin{aligned}&\max_{\varepsilon,t}\ 2(U(t)\varepsilon)^T\Big(U(t)\Big(F(t,u(t)+\varepsilon)-u(t)'\Big)+U(t)'\varepsilon\Big)\\ &\text{s.t. } \Delta_0 \le \|U(t)\varepsilon\|^2 \le \Delta_+,\quad t\in[\underline{t},\bar{t}].\end{aligned}\tag{29}$$

Depending upon the value of Δ_1', we compute bounds for each solution component. If $\Delta_1'>0$, we take $\Delta_+' := \Delta_1'$. Then the definition of Δ_+' implies that

$$\Delta'(t)\le\Delta_+' \qquad \text{if } \Delta(t)\in[\Delta_0,\Delta_+],\qquad t\in[\underline{t},\bar{t}]\tag{30}$$

and we obtains from Theorem 1

$$\Delta(t)\le\overline{\Delta}(t) := \Delta_0+\Delta_1(t-\underline{t})\quad \text{for } t\in[\underline{t},t_1],\tag{31}$$

where

$$t_1=\bar{t},\quad \Delta_1=0,\quad \text{if } \Delta_+'\le 0\tag{32}$$

and

$$t_1=\underline{t}+\frac{\Delta_+-\Delta_0}{\Delta_+'},\quad \Delta_1=\Delta_+',\quad \text{if } \Delta_+'>0.\tag{33}$$

CASE II: *Decaying bound expected.* If we hope for a decaying bound we solve instead the optimization problem

$$\begin{aligned}&\max_{\varepsilon,t}\ 2\Big(U(t)\varepsilon\Big)^T\Big(U(t)\Big(F(t,u(t)+\varepsilon)-u(t)'\Big)+U(t)'\varepsilon\Big)\\ &\text{s.t. } \Delta_- \le \|U(t)\varepsilon\|^2 \le \Delta_0,\quad t\in[\underline{t},\bar{t}].\end{aligned}\tag{34}$$

Now let Δ_2' be the maximum value of (34). Then the definition of Δ_2' implies that

$$\Delta'(t)\le\Delta_2' \qquad \text{if } \Delta(t)\in[\Delta_-,\Delta_0],\qquad t\in[\underline{t},\bar{t}].\tag{35}$$

If $\Delta_2'<0$ then we take $\Delta_1:=\Delta_2'$, and from Theorem 2 we get the bounds of the solution

$$\Delta(t)\le\overline{\Delta}(t) := \Delta_0+\Delta_1(t-\underline{t})\quad \text{for } t\in[\underline{t},t_1],\tag{36}$$

where

$$t_1=\underline{t}+\frac{\Delta_--\Delta_0}{\Delta_-'},\quad \Delta_1=\Delta_-',\quad \text{if } \Delta_-'<0.\tag{37}$$

If $\Delta_2'\ge 0$, we do not get a useful information and must proceed by Case I.

In both cases, if we succeed, then (31) and (36) imply that

$$\Delta(t)\le\overline{\Delta}(t)\quad \text{for } t\in[\underline{t},t_1].\tag{38}$$

Now we see from (26) that

$$\varepsilon(t)=U(t)^{-1}\eta(t),$$

and for any vector c with $V(t) := U(t)^{-T}$, we have

$$\left.\begin{aligned}|c^T\varepsilon(t)| &= |c^TV(t)^T\eta(t)|\\ &= |(V(t)c)^T\eta(t)| \le \|V(t)c\|\|\eta(t)\|,\end{aligned}\right\}$$

by the Cauchy-Schwarz inequality. From (27) and (38) we now find

$$|c^T\varepsilon(t)|\le\|V(t)c\|\Delta(t)^{1/2}\le\|V(t)c\|\overline{\Delta}(t)^{1/2}.\tag{39}$$

In particular, for a unit vector $c=e_k$, we have

$$|\varepsilon_k(t)|\le\|V(t)e_k\|\overline{\Delta}(t)^{1/2}\quad \text{for } t\in[\underline{t},t_1].\tag{40}$$

This inequality gives computable bounds for the componentwise error of the solution.

4 Implementation

This section contributes a description of an overall procedure to compute the error bounds of ODEs. We start with computing the approximate solution u and U for the system of ODEs (1) and the preconditioner (19) for $[0,T]$ by using ODE solver *ode*45. Then we split our problem into pieces at the spline nodes $0=t_0<t_1,\cdots<t_n=T$, and evaluate the system of ODEs and the approximate solutions $u(t_i)$ and $U(t_i)$ at these nodes. For each polynomial piece $[t_i,t_{i+1}]$, we compute the spline coefficients a, b, c, d and A, B, C, D of the system of ODEs and the preconditioner at the bounds t_i, t_{i+1}.

Now we discuss the technique used to compute defect estimates $\overline{\Delta}_l$ and t_l.

We have $0=\underline{t}:=t_i$ as fixed starting time, and $\Delta_0:=n/4$ as starting defect estimate, where n is the dimension of the system of ODEs. We put $\bar{t}:=t_{i+1}$. We begin with the case when bound growth is expected. We need to increase Δ_0 by some factor $q>1$. For $[\underline{t},\bar{t}]$, we solve the optimization problem (29) with $\Delta_+:=q\Delta_0$ and Δ_0. Since we optimize with the local optimization solver IPOPT, we solve the optimization problem (29) with many different starting points and take Δ' as the maximum of all solutions of (29). (This very time-consuming step would not be needed if we would use a reliable global solver.) Depending upon the sign of Δ', we proceed further.

If $\Delta'>0$, we compute t_1 and $\overline{\Delta}(t_1)$ by

$$t_1=\min\Big(\bar{t},\underline{t}+\frac{(\Delta_\pm-\Delta_0)}{\Delta'}\Big),\tag{41}$$

$$\overline{\Delta}(t_1)=\Delta_0+\Delta'(t_1-\underline{t}).\tag{42}$$

There are two possibilities: either $t_1=\bar{t}$ or $t_1<\bar{t}$. If $t_1=\bar{t}$, we take $t_l:=t_1$ and $\overline{\Delta}_l:=\overline{\Delta}(t_1)$ and apply inequality (40)

to compute the component-wise error bounds of ODEs for the polynomial piece $[t_i,t_{i+1}]$. If $t_1 < \bar{t}$, we check whether or not a significant step was taken. If not, then we increase the current value of Δ_+ by a factor qj for $j = 2,\cdots,j_{\max}$, where $j_{\max}$ is the maximum number of attempts we make. With the new value of Δ_+, we solve (29) to compute Δ'. Again, if $\Delta' > 0$, we compute t_1 and the defect estimate $\overline{\Delta}(t_1)$ by using (41) and (42), and check whether $t_1 = \bar{t}$ or t_1 reaches at least 20% of the interval $[\underline{t},\bar{t}]$. If not, we further increase the current value of Δ_+, again solve (29) and continue the above process until we succeed or $j = j_{\max}$. If $j = j_{\max}$ and still we can not reach at least 20% of the interval $[\underline{t},\bar{t}]$, we stop our algorithm, and report that the bounds cannot be extended to larger times.

In case a significant step was taken, we put $t_l := t_1$ and $\overline{\Delta}_l := \overline{\Delta}(t_1)$ and split our interval $[\underline{t},\bar{t}]$ into two subintervals $[\underline{t},t_l]$ and $[t_l,\bar{t}]$. We are done with $[\underline{t},t_l]$. We also take $q_{\text{used}} := q$. For the interval $[t_l,\bar{t}]$, we proceed in the following way.

At $t = t_l$, we compute approximate continuously differentiable solutions u and U for the systems of ODEs and the preconditioner, respectively, and their derivatives u' and U'. Using these values, we compute spline coefficients (24) and (25). We put $\underline{t} := t_l$ and $\Delta_0 := \overline{\Delta}_l$. With the new values of the spline coefficients, $\underline{t}$, Δ_0 and $q := q_{\text{used}}$,where q_{used} is the last value of q used in previous iteration, we again solve the optimization problem (29) to compute Δ'. If $\Delta' > 0$, then we compute t_l and the defect $\overline{\Delta}_l$ by using the same technique as described above. We continue this process of computing t_l, $\overline{\Delta}_l$ until $t_l = \bar{t}$ or $\Delta' \le 0$. If $\Delta' \le 0$, we take $t_l = \bar{t}$, $\overline{\Delta}_l = \Delta_0$ and $q = q_{\text{used}}$. Using these t_l and their corresponding defects $\overline{\Delta}_l$, we apply inequality (40) to compute the error bounds of ODEs for the polynomial piece in $[t_i,t_{i+1}]$.

If $\Delta' < 0$, the bounds are decaying. In that case we decrease Δ_0 by some factor $q > 1$. With new value $\Delta_- := \Delta_0/q$ we solve the optimization problem (34) to compute Δ'.

If $\Delta' \ge 0$, the bounds could not decay further. Therefore, we take $t_l = \bar{t}$ and $\overline{\Delta}_l = \Delta_0$ and compute the error bounds for $[\underline{t},\bar{t}]$ by using (40). If now $\Delta' < 0$, we compute t_1 and $\overline{\Delta}(t_1)$ by (41) and (42).

Now we check whether or not a significant step was taken. If not, we decrease the current value of Δ_- by a factor qj for $j = 2,\cdots,j_{\max}$. With the new value of $\Delta_- := \Delta_0/qj$, we solve (34) to compute Δ'. Again if $\Delta' < 0$, we compute t_1 and the defect estimate $\overline{\Delta}(t_1)$ from (41) and (42), and check whether $t_1 = \bar{t}$ or t_1 reaches at least 20% of the interval $[\underline{t},\bar{t}]$. If not, we further decrease the current value of Δ_- and continue the above process until we succeed or $j = j_{\max}$. If $j = j_{\max}$ and still we can not reach 20% of the interval $[\underline{t},\bar{t}]$, we stop our algorithm, and report that the bounds cannot be extended to larger times.

Otherwise we put $t_l := t_1$ and $\overline{\Delta}_l := \overline{\Delta}(t_1)$, and split our interval $[\underline{t},\bar{t}]$ into two subintervals $[\underline{t},t_l]$ and $[t_l,\bar{t}]$. We are done with $[\underline{t},t_l]$, so we proceed to the interval $[t_l,\bar{t}]$. As far as computing of spline coefficients is concerned, we use the same technique we have described for $\Delta' > 0$. Then we take $\underline{t} := t_l$ and $\Delta_0 := \overline{\Delta}_l$ and new values of spline coefficients, $\underline{t}$, Δ_0 and $q := q_{\text{used}}$, and solve the optimization problem (34) to compute Δ'. If $\Delta' < 0$, we compute t_l and the defect $\overline{\Delta}_l$, using the same technique as described above. We continue this process of computing t_l, $\overline{\Delta}_l$ until $t_l = \bar{t}$ or $\Delta' \ge 0$.

If $\Delta' \ge 0$, we take $t_l = \bar{t}$, $\overline{\Delta}_l = \Delta_0$ and $q_{\text{used}} = q$. Using these t_l and their corresponding defects $\overline{\Delta}_l$, we apply inequality (40) to compute the error bounds of ODEs forthe polynomial piece in $[t_i,t_{i+1}]$.

For the next iteration, $[t_{i+1},t_{i+2}]$, we take $q = q_{\text{used}}$, $\Delta_0 := \overline{\Delta}_l$ and proceed as above.

Now we describe step by step the implementation of our new scheme for computing the error bounds. Each step is graphically represented by a number of items shown in Figure 4.

Step I, items $1-6$**:**
We construct an ODE model in MATLAB consisting of the systems of differential equations (1) and the equation for the preconditioner (19). The initial conditions for this model are chosen as

$$y_0 = \frac{\bar{y}_0 + \underline{y}_0}{2}, \quad Y_0 = \text{Diag}\,(\bar{y}_0 - \underline{y}_0)^{-1}.$$

Using an approximate ODE solver, we compute an approximate solutions of the systems (1) and (19) on a grid determined by the solver. In order to check the error bounds, we also solve (1) for $y(0) = y_1 = \underline{y}_0$ and $y(0) = y_2 = \bar{y}_0$.

Step II, items $7-8$**:**
Using cubic Hermite interpolation, we find an approximate continuously differentiable solution

$$u(t) = u_i(t) \quad \text{for} \quad t_{i-1} \le t < t_i,$$
$$U(t) = U_i(t) \quad \text{for} \quad t_{i-1} \le t < t_i$$

(where $0 = t_0 < t_1 < \cdots < t_m = T$) for the systems (1) and (19) in the form of piecewise cubic polynomials $u(t)$ and $U(t)$ respectively. For each polynomial piece $[t_{i-1},t_i]$ we do Steps III-V.

Step III, items $9-16$**:**
We encode the optimization problems (29) and (34) in AMPL [41] and solve them by feeding the AMPL code into the solver IPOPT [49]. We construct the model file, the data file and the command file to compute the scalar bound function $\overline{\Delta}$.

We take $0 = \underline{t} := t_i$ as fixed starting time and $\Delta_0 := n/4$ (where n is a dimension of the system of ODEs) as the starting defect estimate. We put $\bar{t} := t_{i+1}$ and begin with the case when the bound is expected to increase. We increase Δ_0 by $\Delta_+ := q\Delta_0$, where $q > 1$ is fixed, and solve the optimization problem (29) for $[\underline{t},\bar{t}]$ and $[\Delta_0,\Delta_+]$.

Step IV, items $17,18,25-29$**:**
If the result gives $\Delta' \le 0$, this means that the bound is decaying. Therefore, we solve the optimization problem (34)

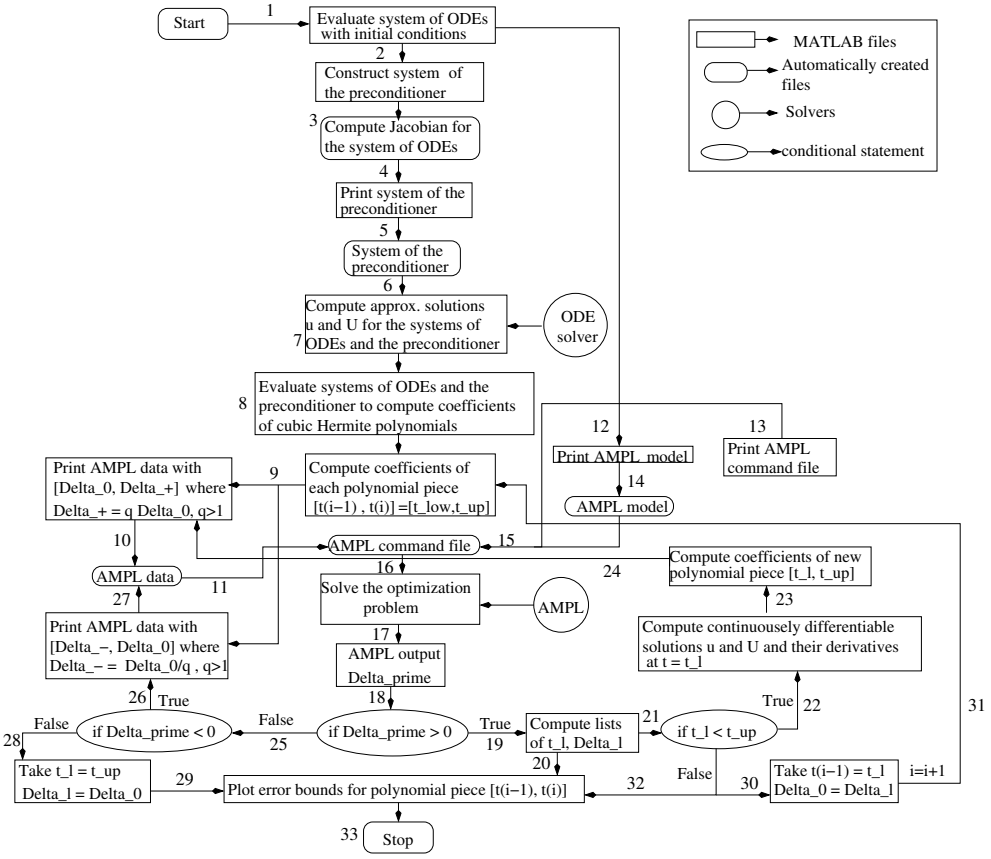


Fig. 1 DIBIS: a graphical representation

for $[\underline{t},\bar{t}]$ and $[\Delta_-,\Delta_0]$, $\Delta_- = \Delta_0/q$. Since we are doing local optimization but need a global optimum, we solve these optimization problems with many different starting points, and take Δ' as the maximum of all solutions of either (29) or (34).

Step V, items $19-21$**:**
Depending upon the sign of Δ', we compute t_1 and $\overline{\Delta}(t_1)$ by (41) and (42).

There are two possibilities: either $t_1 = \bar{t}$ or $t_1 < \bar{t}$. If $t_1 = \bar{t}$, then we take $t_l := t_1 = \bar{t}$ and $\overline{\Delta}_l := \overline{\Delta}(t_1)$ and go to Step VII, otherwise we increase/decrease the current value of $\Delta_\pm$ by a factor qj, $j = 1,\cdots,j_{\max}$ and we solve the optimization problem (29) with $\Delta_+ := qj\Delta_0$, or the optimization problem (34) with $\Delta_- := \Delta_0/qj$. We repeat Steps IV–V to compute t_l and $\overline{\Delta}_l$.

We continue until a significant step has been taken, that is t_1 reaches at least 20% of the interval $[\underline{t},\bar{t}]$.

Step VI, items $22-24$**:**
We put $t_l := t_1$ and $\overline{\Delta}_l := \overline{\Delta}(t_1)$ and split our interval $[\underline{t},\bar{t}]$ into two subintervals $[\underline{t},t_l]$ and $[t_l,\bar{t}]$. We are done with $[\underline{t},t_l]$. We also take $q_{\text{used}} := q$. For the interval $[t_l,\bar{t}]$, we take $t := t_l$, $\Delta_0 := \Delta_l$ and $q := q_{\text{used}}$.

We repeat Steps II–VI with $l = l+1$.

We continue until $l = l_{\max}$ and $t_{l_{\max}} = \bar{t}$.

Step VII, items $32-33$**:**
Using lists of t_l and $\overline{\Delta}_l$ computed for $[t_i,t_{i+1}]$, we plot the componentwise error in the solution by using inequality (40). Now we take $\Delta_0 := \overline{\Delta}_l$ and $\underline{t} := t_l$ and return to Step II for the next iteration, $i = i+1$.

Remarks. 1. There is a possibility that $t_{l_{\max}} < \bar{t}$ in Step VI. Therefore, if after finite many iterations say $j = j_{\max}$ in Step V, the goal that t_1 reaches at least 20% of the interval $[\underline{t},\bar{t}]$ is not achieved, we stop the program, and report that the bounds cannot be extended to larger times.

2. Since in the current implementation we are only doing multiple start local optimizations, the bounds obtained are not rigorous. In order to get rigorous bounds, one would need to do instead rigorous global optimization with rounding error control.

Graphical representation

A graphical representation of our method DIBIS for enclosing the solutions is given by the following flow diagram.

Full details are given in the first author's Ph.D. thesis [12]. In particular, Appendix B in [12] contains a MATLAB routine driver.m for how to call DIBIS, and further explanations of Figure 1. The whole package is automatized. User's

input consists just of the system of ODEs with initial conditions.

5 Numerical results

In this section we apply our solver DIBIS to compute error bounds for some sample ODEs. The results obtained from DIBIS are compared with the solvers VALENCIA-IVP, V-NODE-LP, and VSPODE, for which we were able to obtain implementations.

The results for DIBIS were usually produced with a large step size of $h = 0.1$, and in two cases $h = 0.01$ was needed. The other solvers needed much smaller step sizes. As we shall see, the results obtained from VALENCIA-IVP are not acceptable for the step size $2e-4$. In order to produce tight enclosures for VNODE-LP, the order of method is kept between 5 and 8 with step size $1e-5$ and in some cases even $1e-7$. For higher orders, VNODE-LP does not behave well and diverges at very early stages. To get good results for VSPODE, the orders of the Taylor model and the interval Taylor series method were chosen as 17 or 18 depending upon the nature of the problem, with step size $1e-3$.

The reported widths $W_i = y_{\sup} - y_{\inf}$, $i = 1, \cdots, n$, are evaluated at t_{end}. In cases where a solver did not reach t_{end}, we usually tried several variants for the solver settings, and reported the best results.

To get an idea of the minimal size of the solution set, we calculated approximate solutions for initial values at the lower bound, the midpoint, and the the upper bound of the initial box. (For a better inner approximation, one would have to use a Monte-Carlo approach.)

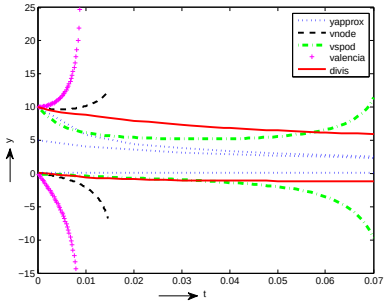
5.1 Example 1

$$y' = -y^3,$$
$$t \in [0,.07], \quad y(t_0) = y_0 \in [0.1, 10].$$

Method	$W(t=.07)$	$W(t=1)$	$W(t=100)$	CPU time(sec)
VALENCIA-IVP	–	–	–	–
VNODE-LP	–	–	–	–
VSPODE	21.13	–	–	0.0396
DIBIS	5.28	2.75	0.30	9.975
y_{approx}	2.48×10^{-2}	0.61	0.01	–

Table 1 Results of Example 1

VALENCIA-IVP and VNODE-LP do not reach t_{end}. V-SPODE reaches t_{end}; the width of the enclosure W computed at t_{end} is 21.1296 with order of Taylor model 18 and order of interval Taylor series method 17. DIBIS reaches t_{end}; the



best solution for correct printing. another method is scaling with scale=0.3

Fig. 2 Enclosures with $q = 1.05$, $\delta_0 = 1e-4$, $\delta_1 = 1e-4$, $h = .01$, $k = 1$

width of the enclosure at $t_1 = t_{\text{end}}$ is $W = 5.2849$. DIBIS computes much tighter enclosures as t increases from $t = .07$ to $t = 1$ and $t = 100$.

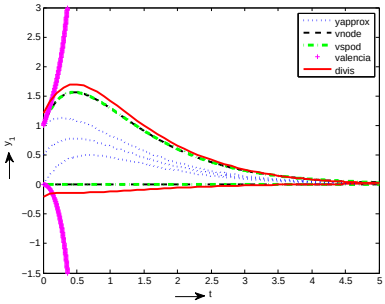
VALENCIA-IVP diverges at $t_1 = 0.0092$; the width of the enclosure computed at t_1 is $W = 5.0499e+03$. VNODE-LP gives up at $t_1 = 0.01468$ with step size $1e-7$; the width of the enclosure computed at t_1 is $W = 18.9957$ with order of method $p = 8$.

5.2 Example 2

$$y_1' = y_1 - 2y_2,$$
$$y_2' = 3y_1 - 4y_2,$$
$$t \in [0,5], \; y(t_0) \in [0,1] \times [-1,0].$$

VALENCIA-IVP stops at $t_1 = 1.2388$ with wide enclosures; $W_1 = 4.5945 \times 10^2$, $W_2 = 9.9963 \times 10^2$.

The enclosures computed by VNODE-LP, and VSPODE at t_{end} are almost the same upto 4 decimal places. They are better than those of DIBIS, again due to the unavoidable wrapping by the initial ellipsoid. The order of the method for VNODE-LP is kept at $p = 5$ with the step size $1e-5$. The order of the Taylor model and the interval Taylor series method for VSPODE are 18 and 17, respectively.



solution for correct printing. Another method is scaling with scale=0.3

Fig. 3 Enclosures with $q = 1.001$, $\delta_0 = 1e-4$, $\delta_1 = 1e-4$, $h = .1$, $k = 1$

Method	$W_1(t_{\text{end}})$	$W_2(t_{\text{end}})$	CPU time(sec)
VALENCIA-IVP	–	–	–
VNODE-LP	3.3508×10^{-2}	3.3417×10^{-2}	2.77
VSPODE	3.3508×10^{-2}	3.3417×10^{-2}	2.10541
DIVIS	5.71×10^{-2}	5.621×10^{-2}	39.50
y_{approx}	6.7379×10^{-3}	6.7379×10^{-3}	–

Table 2 Results of Example 2

5.3 Example 3

This is an example of chemical reaction dynamics taken from [15].

$$y_1' = y_1^2 y_2^2 - y_1^4 - y_1 y_2^2 + y_1^3 - y_2^4,$$
$$y_2' = y_1 y_2^3 - y_1^3 y_2 - y_2^3 + y_1^2 y_2,$$
$$t \in [0, .35],\ y(t_0) \in [0.2, 1] \times [0.6, 1].$$

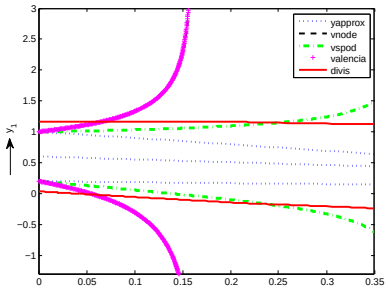
Method	$W_1(t_{\text{end}})$	$W_2(t_{\text{end}})$	CPU time(sec)
VALENCIA-IVP	–	–	–
VNODE-LP	–	–	–
VSPODE	2.119	1.124	4.44
DIBIS	0.658	0.659	44.92
y_{approx}	0.494	0.426	–

Table 3 Results of Example 3

VALENCIA-IVP computes enclosures until $t_1 = .1608$ with wide bounds; $W_1 = 1.6349 \times 10^1$, $W_2 = 1.1209 \times 10^1$.

VNODE-LP (with order $p = 5$) reaches only $t_1 = .1558$ with wide bounds; $W_1 = 5.1006$, $W_2 = 3.6407$. VSPODE reaches t_{end}, but the enclosures are again wide. The order of Taylor model and interval Taylor series method are 18 and 17, respectively.

DIBIS computes much tighter enclosures at t_{end}.



best
solution for correct printing. another method is scaling with scale=0.3

Fig. 4 Enclosures with $q = 1.001$, $\delta_0 = 1e-4$, $\delta_1 = 1e-4$, $h = .1$, $k = 1$

5.4 Example 4

This is another example of a chemical reaction with different chemistry and is taken from [15].

$$y_1' = -2y_2 + y_2 y_3 - y_1^3,$$
$$y_2' = y_1 - y_1 y_3 - y_2^3,$$
$$y_3' = y_1 y_2 - y_3^3,$$
$$t \in [0, .549858],\ y(t_0) \in [0, 1.5] \times [0, 1.2] \times [0, 1].$$

VALENCIA-IVP computes enclosures until $t_1 = .2576$ with wide bounds $W_1 = 4.0314e+04$, $W_2 = 8.5279$, $W_3 = 5.7964$.

VNODE-LP computes enclosures until $t_1 = .2520$ with better bounds $W_1 = 7.9141$, $W_2 = 4.7274$, $W_3 = 3.3217$ with

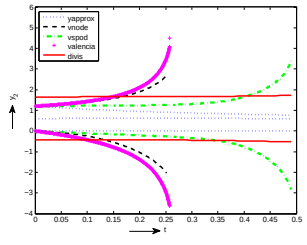


Fig. 5 Enclosures with $q = 1.01$, $\delta_0 = 1e-4$, $\delta_1 = 1e-4$, $h = .01$, $k = 1$

order of method $p = 5$, but nevertheless couldn't continue further.

The enclosures computed by VSPODE with order of Taylor model = 17 and order of interval Taylor series method = 17 diverges after t_{end} (which is why we chose this t_{end}).

DIVIS reached t_{end} and can be continued beyond this value of t.

Method	$W_1(t_{\text{end}})$	$W_2(t_{\text{end}})$	$W_3(t_{\text{end}})$	CPU time(sec)
VALENCIA-IVP	–	–	–	–
VNODE-LP	–	–	–	–
VSPODE	7.35	3.05	3.21	2.67
DIVIS	3.22	1.72	1.99	–
y_{approx}	5.14×10^{-1}	7.48×10^{-1}	9.14×10^{-1}	55.45

Table 4 Results of Example 4

Discussion. It can be observed from the above presented examples that DIBIS computes the best enclosures among all methods tested for ODEs or systems of ODEs containing higher order polynomials. This is also the case in other highly nonlinear test problems.

We only need to compute the Jacobian of the system of ODEs to construct the preconditioner, whereas the other solvers need high derivatives and the resulting Taylor series are not so well-behaved. Since our defect estimates are computed by using local optimization with many starting points, the execution time is significantly larger, though.

On less nonlinear examples, the performance of the solvers is more similar. However, since we are using outer ellipsoidal approximation, our initial enclosing set is already an ellipsoid enclosing the initial box, which leads to an unavoidable increase in the volume of the enclosures. As a result, our bound are slightly worse than those of the other solvers on simple problems where the latter are highly accurate. (Things would be opposite when the initial uncertainty is ellipsoidal; in this case the other methods would have this disadvantage.)

We also note that our results are not fully rigorous since we use a heuristic global optimization solver (multiple local search) only. The rigorous results would usually be only slightly worse than these heuristic results because wide initial intervals dominate the errors, and the additional effect of the rounding errors is minimal. Only in the rare case where the multistart heuristics fails to find the global minimum, the true bounds could be significantly worse. (This is why we used a large number of starting points.) Rigorous bounds could be found by replacing our optimization heuristics with a rigorous global solver with full error control together with interval arithmetic toolbox IntLab [46].

6 Perspectives and future work

We have developed a theory to compute validated enclosures for the solutions of systems of ODEs with uncertain initial conditions. The proposed method computes defect estimates by using optimization techniques. Then by applying differential inequality, validated state enclosures for IVPs are computed that compare favorably with the existing solvers VALENCIA-IVP, VNODE-LP and VSPODE.

In the future, we intend to improve the efficiency of this scheme. Instead of using the local optimizer IPOPT [49] with multiple starts, we will use a rigorous global optimization solver to make our bounds fully rigorous. Preliminary results with the COCONUT solver showed that a warm start facility will be needed to solve the many similar global optimization problems in a reasonable time.

At present, we are using cubic Hermite spline to approximate the solution. If the solution is highly differentiable,

the error can be reduced further by using higher-order spline approximations.

At present, step length and regularizing parameters are chosen by user. We intend to automatize their choice. This would involve looking into the nature of the problem by inspecting the Jacobian of the system, condition numbers etc..

The theory also applies to the systems of ODEs with uncertain parameters, and we intend to extend the implementation in this direction.

References

1. E. Adams. *On methods for the construction of the boundaries of sets of solutions for the differential equations or finite-dimensional approximations with input sets*. Bericht. Fakultät für Mathematik. Universität Karlsruhe, 1979. http://books.google.at/books?id=PVgRHAAACAAJ.
2. C. Bendtsen and O. Stauning. Fadbad, a flexible c++ package for automatic differentiation. *Department of Mathematical Modelling, Technical University of Denmark*, 1996.
3. C. Bendtsen and O. Stauning. Tadiff, a flexible c++ package for automatic differentiation. *TU of Denmark, Department of Mathematical Modelling, Lungby. Technical report IMM-REP-1997-07*, 1997.
4. M. Berz. COSY INFINITY Version 8 referenced manual. Technical Report MSUCL–1088, National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, 1997. http://bt.pa.msu.edu/pub/papers/cpo5cosy/cpo5cosy.pdf.
5. F. L. Chernousko. *State Estimation for Dynamic System*. CRC Press, 1993.
6. F. L. Chernousko. *What is ellipsoidal modeling and how to use it for control and state estimation?* Springer, Wien, 1999.
7. F. L. Chernousko. Ellipsoidal state estimation for dynamical systems. *Nonlinear Analysis: Theory, Methods & Applications*, 63(57):872–879, 2005. http://www.sciencedirect.com/science/article/pii/S0362546X05000210.
8. F. L. Chernousko. Properties of optimal ellipsoids approximating reachable sets of uncertain systems. *Mathematical and Computer Modeling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences*, 11:135–147, 2005.
9. S. Dietrich. *Adaptive verifizierte Lösung gewöhnlicher Differentialgleichungen*. Universität Karlsruhe (TH), Germany, 2003. PhD Thesis.
10. I. Eble. *Über Taylor-Modelle*. Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology (KIT) D-76128 Karlsruhe, 2007. http://iamlasun8.mathematik.uni-karlsruhe.de/~ae08/dissertation/Dissertation.pdf.
11. P. Eijgenraam. *The solution of initial value problems using interval arithmetic*. PhD thesis, Stichting Mathematisch Centrum, Amsterdam, 1981.
12. Q. Fazal. *Optimization techniques for error bounds of ODEs*. PhD thesis, Faculty of Mathematics, University of Vienna, Austria, 2011. http://www.mat.univie.ac.at/~neum/ms/qaisraPhD.pdf.
13. T.F. Filippova. Reachable sets of a nonlinear impulsive control system under uncertainty. In *7th European Nonlinear Dynamics Conference (ENOC)*, Rome, Italy, July 2011. http://w3.uniroma1.it/dsg/enoc2011/proceedings/pdf/Filippova.pdf.
14. T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, 20:292–296, 1919. http://www.jstor.org/stable/1967124.
15. M. Janssen, P. V. Hentenryck, and Y. Deville. Optimal pruning in parametric differential equations, 2001. International Conference on Constraint Programming (CP-2001).
16. O. Knüppel. PROFIL/BIAS a fast interval library. *Computing*, 53:277–287, 1994. http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html.
17. W. Kuhn. Rigorous error bounds for IVP based on defect estimates, 1997.
18. A.B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for reachability analysis. Part I: External approximations. *Optimization Methods and Software*, 17(2):177–206, 2002. http://www.tandfonline.com/doi/pdf/10.1080/10556780210000012426.
19. A.B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for reachability analysis. Part II: Internal approximations box-valued constraints. *Optimization Methods and Software*, 17(2):207–237, 2002. http://www.tandfonline.com/doi/pdf/10.1080/10556780210000012435.
20. A.B. Kurzhanski and P. Varaiya. Reachability under state constraints:the ellipsoidal technique. In *Proc. of 15th Triennial IFAC World Congress, Barcelona, Spain*, 2002. http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2002/data/content/01408/1408.pdf.
21. K. l. Nickel. Bounds for the set of solutions of functional-differential equations. *Annales Polonici Mathematici*, 42:241–257, 1983.
22. V. Lakshmikantham and S. Leela. *Differential and Integral Inequalities: Theory and Applications: Vol.: 1: Ordinary Differential Equations*. Academic Press, 1969.
23. Y. Lin and M. A. Stadther. Validated solution of ODEs with parametric uncertainties. *Computer-Aided Chem. Engrg*, 21:167–172, 2006.
24. Y. Lin and M. A. Stadther. Validated solutions of initial value problems for parametric ODEs . *Applied Numerical Mathematics*, 57:1145–1162, 2007.
25. R. J. Lohner. Enclosing the solution of ordinary initial and boundary value problems. *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286, March 28-April 1 1987.
26. R. J. Lohner. *Step size and order control in the verified solution of IVP with ODEs*, March 28-April 1 1995.
27. RJ Lohner. Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In *INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS CONFERENCE SERIES*, volume 39, pages 425–425. OXFORD UNIVERSITY PRESS, 1992.
28. K. Makino and M. Berz. COSY INFINITY version 8. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 427:338–343, 1999.
29. K. Makino and M. Berz. *Taylor model based verified integration for the Volterra equations and the Lorenz system*, May 23-25 2002. SIAM Workshop on Validated Computing Toronto.
30. K. Makino and M. Berz. COSY INFINITY version 9. In *Proceedings of the 8th International Computational Accelerator Physics Conference - ICAP 2004*, volume 558, pages 346–350, 2006.
31. R. E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. *Error in Digital Computation*, 1:61–130, 1965. http://interval.louisiana.edu/Moores_early_papers/Moore_in_Rall_V1.pdf.
32. R. E. Moore. Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solution of ordinary differential equations. *Error in Digital Computation*, 2:103–140, 1965. http://interval.louisiana.edu/Moores_early_papers/Moore_in_Rall_V2.pdf.

33. R. E. Moore. Interval analysis. Englewood Cliffs, N. J., 1966.
34. R.E. Moore, J.A. Davison, H.R. Jaschke, S. Shayer, LOCKHEED MISSILES, and SPACE CO PALO ALTO CALIF LOCKHEED PALO ALTO RESEARCH LAB. *DIFEQ INTEGRATION ROUTINE-USER'S MANUAL*. Defense Technical Information Center, 1964. http://books.google.at/books?id=zruONwAACAAJ.
35. N. S. Nedialkov. Interval Tools for ODEs and DAEs. In *In: CD-Proc. of the 12th GAMMIMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006.*, IEEE Computer Society, Los Alamitos (2007), 2006.
36. Nedialko S. Nedialkov. Implementing a rigorous ode solver through literate programming. In Andreas Rauh and Ekaterina Auer, editors, *Modeling, Design, and Simulation of Systems with Uncertainties*, volume 3 of *Mathematical Engineering*, pages 3–19. Springer Berlin Heidelberg, 2011. http://dx.doi.org/10.1007/978-3-642-15956-5_1.
37. N.S. Nedialkov. *Computing Rigorous Bounds on the Solution of an IVP for an ODE*. PhD thesis, University of Toronto, 1999. http://www.cs.toronto.edu/NA/reports.html$\sharp$ned-99-phd.
38. N.S. Nedialkov and K.R. Jackson. An Interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an IVP for an ODE. *Reliable Computing*, 5(3):289–310, 1998.
39. A. Neumaier. Global, rigorous and realistic bounds for the solution of dissipative differential equations. Part I: Theory. *Computing*, 52:315–336, 1994. www.mat.univie.ac.at/~neum/papers.html.
40. I. Papamichail and C. S. Adjiman. A rigorous global optimization algorithm for problems with ordinary differential equations. *J. of Global Optimization*, 24(1):1–33, September 2002.
41. D.M. Gay R. Fourer and B.W. Kernighan. *AMPL: a modeling language for mathematical programming*. Brooks/Cole Publishing Company/Cengage Learning, 2003. http://ampl.com/?gclid=CJ245OXp5LECFUHxzAodby0AYg,.
42. A. Rauh, E. Auer, and E. P. Hofer. A novel interval method for validating state enclosures of the solution of initial value problems. *International Journal of Applied Mathematics and Computer Science*, 19(3):381–397, 2008. http://versita.metapress.com/content/858x533532847056/fulltext.pdf.
43. A. Rauh, E. Auer, J. Minisini, and Eberhard P. Hofer. Extensions of ValEncIA-IVP for reduction of overestimation, for simulation of differential algebraic systems, and for dynamical optimization. *PAMM*, 7(1):1023001–1023002, 2007. http://onlinelibrary.wiley.com/doi/10.1002/pamm.200700022/pdf.
44. A. Rauh, E.P. Hofer, and E. Auer. VALEncIA-IVP: A comparison with other initial value problem solvers. In *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006. 12th GAMM - IMACS International Symposium on*, page 36, sept. 2006. http://www.valencia-ivp.com.
45. A. Reiter. *INTERVAL ARITHMETIC PACKAGE (INTERVAL) FOR THE CDC 1604 AND CDC 3600*. WISCONSIN UNIV MADISON MATHEMATICS RESEARCH CENTER, 1968.
46. S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. http://www.ti3.tu-harburg.de/rump/.
47. H. Schichl. Global optimization in the coconut project. *Numerical Software with Result Verification*, pages 277–293, 2004. http://www.mat.univie.ac.at/~neum/glopt/coconut/.
48. O. Stauning. *Automatic Validation of Numerical Solutions*. PhD thesis, IMM-DTU, September 1997. http://chaos.unh.edu/~kouroshz/scientific_computing/assignment/int_arith.pdf, http://www2.imm.dtu.dk/~km/FADBAD/.
49. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
50. W. Wolfgang. *Differential and integral inequalities. Translated by Lisa Rosenblatt and Lawrence Shampine*. Springer-Verlag, Berlin, New York, 1970.