AN INTERVAL METHOD FOR INITIAL VALUE PROBLEMS IN LINEAR ORDINARY DIFFERENTIAL EQUATIONS*

NEDIALKO S. NEDIALKOV†

Abstract. Interval numerical methods for initial value problems (IVPs) for ordinary differential equations (ODEs) compute bounds that contain the solution of an IVP for an ODE. This paper develops and studies an interval method for computing bounds on the solution of a linear ODE. Our method is based on enclosing the terms in the formula for the closed form solution using Taylor series and various interval techniques. We also present an alternative approach, for bounding solutions, based on following vertices that specify a parallelepiped enclosing the solution of a linear ODE.

We propose a simple combination of two existing methods—the parallelepiped and QR-factorization methods—for reducing the wrapping effect, to obtain a better method for reducing it. The resulting method computes bounds than are as tight as the bounds produced by the QR-factorization method and often much tighter.

Key words. interval methods, Taylor series, initial value problems, linear ordinary differential equations, wrapping effect

AMS subject classifications. 65L05, 65G20, 65L70, 41A58

1. Introduction. Interval numerical methods for IVPs for ODEs [4, 13, 16, 20, 24, 27] are usually designed for problems in the general form $y' = f(y)$, $y(t_0) \in \boldsymbol{y}_0$. These methods compute bounds that contain the solution to this problem at points in a given interval $[t_0, t_{\text{end}}]$ for all $y(t_0) \in \boldsymbol{y}_0$. Here, f is normally required to be sufficiently differentiable, and $\boldsymbol{y}_0$ is an interval vector, or box, specifying a set of initial conditions. (This autonomous form is not a restriction of consequence, since a time-dependent system of ODEs can be converted into an autonomous system.) Interval methods employ interval arithmetic [15] and various interval techniques to enclose rounding and truncation errors, thus ensuring that all errors in a computation are included in the computed bounds.

To date, interval methods designed specifically for linear ODEs have not been investigated in the literature. Such a method should be easier to describe, implement, and study than a general-type method. Furthermore, a method suitable for linear ODEs should facilitate a better understanding of interval methods for IVPs for ODEs, and in particular, of the *wrapping effect* [15], which has been one of the main obstacles in computing tight bounds on the solution of an IVP for an ODE. The wrapping effect occurs when a solution set that is not a box is enclosed, or wrapped by a box, thus introducing an overestimation on the solution set. By performing a wrapping on each integration step, these overestimations can quickly accumulate, leading to unacceptably wide bounds—the wrapping effect.

In this paper, we develop and investigate a method for computing bounds on the solution of the linear IVP

$$y' = A(t)y + g(t), \quad y(t_0) = y_0$$

for all $y_0 \in \boldsymbol{y}_0$. Here A and g are matrix- and vector-valued functions. Our method is based on enclosing the terms in the formula for the closed form solution to this IVP using Taylor series.

*This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

†Department of Computing and Software, McMaster University, Hamilton, Ontario, L8S 4L7, Canada. email: nedialk@mcmaster.ca

We illustrate the wrapping effect and two anti-wrapping strategies, namely, the parallelepiped [7, 13] and QR-factorization [13] methods for reducing the wrapping effect. The latter has been the most successful, general method for reducing it; see also [18]. We give some insights into the wrapping effect, which, to the author's knowledge, have not been discussed in the literature. As an outcome of these insights, a simple combination of the parallelepiped and QR-factorization methods is proposed. The resulting method computes bounds that are as tight as the bounds produced by each of these methods, and often much tighter than the bounds resulting from the QR-factorization approach.

Another major source of overestimations in interval methods is the *dependency problem*. It arises when dependent variables are treated independently in interval expressions, which is frequently the case in interval computations. Because of this problem, computing tight bounds on a solution, when the initial condition set is not sufficiently small, has been a challenging problem. Except for the Taylor model approach [4], which handles the dependency problem effectively, interval methods for IVPs for ODEs generally cannot compute tight bounds when the initial condition set is not sufficiently small, or when the solution set does not remain small enough.

In our method, the dependency problem is minimal—it occurs only when enclosing truncation errors. In practice, the bounds on truncation errors are kept small by satisfying some user-specified tolerance. Hence, we do not impose restrictions on the size of the initial condition set. Moreover, when studying the wrapping effect on linear problems, one can neglect overestimations due to the dependency problem.

We also outline a second approach for computing bounds on the solution of a linear ODE. This approach is based on "following" vertices that specify a parallelepiped enclosing the solution [12, 22]. However, the resulting method has subtle drawbacks. We discuss them in this paper.

In the remainder of this introduction, we present the problem that is the subject of this paper, a short background on interval methods for IVPs for ODEs, and an outline of this paper.

1.1. The problem. We consider

$$y' = A(t)y + g(t), \tag{1.1}$$

$$y(t_0) = y_0 \in \boldsymbol{y_0}. \tag{1.2}$$

Here, $t \in [t_0, t_{\text{end}}]$ for some $t_{\text{end}} > t_0$, $t_0, t_{\text{end}} \in \mathbb{R}$; A and $g \in C^{(p-1)}(D)$, $p \geq 1$, are real matrix- and vector-valued functions, respectively; and $[t_0, t_{\text{end}}] \subseteq D \subseteq \mathbb{R}$. In our method, p is the order of the truncation error. In (1.2), $\boldsymbol{y}_0$ is an interval vector, which allows specifying a set of initial values, not just a point initial condition (as in standard numerical methods for IVPs for ODEs). The condition $t_{\text{end}} > t_0$ is for expositional convenience—the techniques described in this paper can be readily adapted for $t_{\text{end}} < t_0$.

We wish to compute interval vectors that are guaranteed to contain the solution of (1.1–1.2), for all $y_0 \in \boldsymbol{y_0}$, at points $t_0 < t_1 < \cdots < t_m = t_{\text{end}}$. That is, our goal is to find an interval vector $\boldsymbol{y_j}$ at each t_j, such that

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_j.$$

Here, $y(t; t^*, v)$ denotes the solution to (1.1) with an initial condition $y(t^*) = v$, and $y(t; t^*, \boldsymbol{v}) = \big\{y(t; t^*, v) \mid v \in \boldsymbol{v}\big\}$.

1.2. Background. Interval methods for ODEs are typically based on Taylor series [4, 7, 13, 15, 16, 24, 27], where automatic differentiation (AD) [8, 15, 23] is employed to compute Taylor coefficients. One reason for the popularity of the Taylor series approach is the simple form of the error term, which can be readily bounded using AD. In addition, the order of a Taylor series method and its stepsize can be easily changed from step to step.

The interval Hermite-Obreschkoff (IHO) method [16, 17], as a generalization of interval Taylor series methods, also shares the above characteristics. It is shown in [16, 17] that, for the same stepsize and order, the IHO method is more accurate and efficient than an interval Taylor series method.

The above are one-step methods, where the step from t_{j-1} to t_j consists of the following two phases.

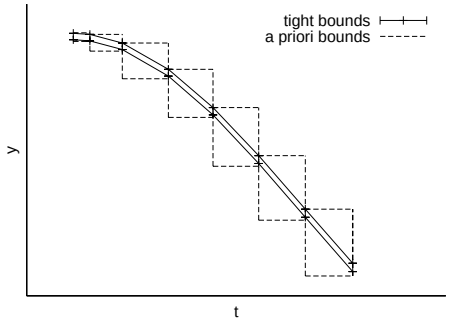


FIG. 1.1. *An illustration of a priori and tight bounds. The tight bounds at different t are connected with straight lines.*

Algorithm I validates existence and uniqueness of the solution in $[t_{j-1}, t_j]$ and computes an a priori enclosure $\widetilde{\boldsymbol{y}}_j$ such that

$$y(t; t_{j-1}, \boldsymbol{y}_{j-1}) \subseteq \widetilde{\boldsymbol{y}}_j, \quad \text{for all } t \in [t_{j-1}, t_j];$$

see Figure 1.1; and

Algorithm II computes a tighter enclosure $\boldsymbol{y}_j \subseteq \widetilde{\boldsymbol{y}}_j$ such that

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_j.$$

(In the above, $y(t; t^*, v)$ refers to the solution of the general $y' = f(y)$ with an initial condition $y(t^*) = v$, and $y(t; t^*, \boldsymbol{v}) = \{y(t; t^*, v) \mid v \in \boldsymbol{v}\}$.)

Janssen et al. [9] develop a constraint satisfaction approach for enclosing the solution to an IVP for an ODE. They introduce a pruning component to an interval method to reduce the computed enclosures by eliminating regions that do not contain a solution. Theoretical and numerical results reported in [9] show that their constraint satisfaction method is more efficient than the IHO approach. They also present a multistep method, in which pruning is performed over enclosures on the solution at several integration points.

All of the above methods employ Taylor expansions with respect to time only. As a consequence, they generally work well with point initial conditions or interval initial conditions, when the intervals are sufficiently small. Berz and Makino's Taylor models [4] employ Taylor series expansions in both time and the initial conditions.

Their approach is particularly suitable for integrating nonlinear ODEs, when the initial conditions are contained in intervals that are not necessarily very small.

The methods in [9, 13, 19, 27] require the computation of Taylor coefficients not only for the solution of an ODE, but also for the solution of its variational equation. Hence, AD facilities for computing these coefficients are needed. For example, in the VNODE [19] and ADIODES [27] packages, TADIFF [3] is used to compute Taylor coefficients for the solution to the given ODE, and TADIFF combined with FADBAD [2] are employed to compute Taylor coefficients for the associated variational equation. An important advantage of our method is that it requires only the computation of Taylor coefficients for the solution of (1.1). As a consequence, implementing such a method becomes less difficult than implementing a general-purpose method.

1.3. Outline. Section 2 gives a brief introduction to interval arithmetic and discusses computing Taylor coefficients, existence and uniqueness of a solution to a linear ODE, and a Taylor series method for enclosing the solution to an ODE with a point initial condition. Our interval method is derived in section 3. Section 4 illustrates the wrapping effect and how the parallelepiped and QR-factorization methods behave. Then, a combination of these two methods is proposed. Section 5 considers measuring the overestimations in our method, namely, the local and global excess. Section 6 gives a simple strategy for stepsize control. Numerical results are reported in section 7. Section 8 outlines an alternative interval method for linear problems and elaborates on its drawbacks. Concluding remarks are given in the final section 9.

2. Preliminaries.

2.1. Interval arithmetic. We introduce interval arithmetic and present some of its properties that we use later. For a more detailed exposition on interval arithmetic, see for example [1] or [15]. In this paper, we use bold font to denote intervals, interval vectors, and interval matrices. We use small letters for scalars, intervals, vectors, and interval vectors. We use capital letters for matrices and interval matrices.

An interval $\boldsymbol{a} = [\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}]$ is the set of real numbers

$$\boldsymbol{a} = [\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}] = \left\{ x \mid \underline{\boldsymbol{a}} \le x \le \overline{\boldsymbol{a}}, \quad \underline{\boldsymbol{a}}, \overline{\boldsymbol{a}} \in \mathbb{R} \right\}.$$

We shall denote the set of real intervals by $\mathbb{IR}$. If $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{IR}$, and $\circ \in \{+, -, \cdot, /\}$, then the interval-arithmetic operations are defined by

$$\boldsymbol{a} \circ \boldsymbol{b} = \left\{ x \circ y \mid x \in \boldsymbol{a},\ y \in \boldsymbol{b} \right\}, \quad 0 \notin \boldsymbol{b} \text{ when } \circ = / \tag{2.1}$$

[15]. This definition can be written in the following equivalent form (we omit $\cdot$ in the notation):

$$\boldsymbol{a} + \boldsymbol{b} = [\underline{\boldsymbol{a}} + \underline{\boldsymbol{b}}, \overline{\boldsymbol{a}} + \overline{\boldsymbol{b}}], \tag{2.2}$$

$$\boldsymbol{a} - \boldsymbol{b} = [\underline{\boldsymbol{a}} - \overline{\boldsymbol{b}}, \overline{\boldsymbol{a}} - \underline{\boldsymbol{b}}], \tag{2.3}$$

$$\boldsymbol{a}\boldsymbol{b} = [\min\left\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\right\}, \max\left\{\underline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \underline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\underline{\boldsymbol{b}}, \overline{\boldsymbol{a}}\,\overline{\boldsymbol{b}}\right\}], \tag{2.4}$$

$$\boldsymbol{a}/\boldsymbol{b} = [\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}][1/\overline{\boldsymbol{b}}, 1/\underline{\boldsymbol{b}}], \quad 0 \notin \boldsymbol{b}. \tag{2.5}$$

Formulas (2.2–2.5) allow us to express the interval arithmetic operations using the endpoints of the interval operands, thus obtaining formulas for real interval arithmetic. In (2.1, 2.5), division by an interval containing zero is not defined. Recent developments have extended interval arithmetic with operations including division by such intervals and operations with infinities [11, 30].

The strength of interval arithmetic when implemented on a computer is in computing rigorous enclosures of real operations by including rounding errors in the computed bounds. To include such errors, we round the resulting (real) intervals in (2.2–2.5) outwards, thus obtaining machine interval arithmetic. For example, when adding intervals, we round $\underline{\boldsymbol{a}}+\underline{\boldsymbol{b}}$ down and round $\overline{\boldsymbol{a}}+\overline{\boldsymbol{b}}$ up. To simplify the discussion, we assume real interval arithmetic in this paper. Because of the outward roundings, intervals computed in machine interval arithmetic always contain the corresponding real intervals.

Definition (2.1) and formulas (2.2–2.5) can be extended to vectors and matrices. If their components are intervals, we obtain interval vectors and matrices. We denote the set of n-dimensional real interval vectors by $\mathbb{IR}^n$ and the set of $n \times m$ real interval matrices by $\mathbb{IR}^{n\times m}$. The arithmetic operations involving interval vectors and matrices are defined by the same formulas as in the scalar case, except that scalars are replaced by intervals.

For example, if $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{IR}^n$, the components of $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{b}$ are given by

$$\boldsymbol{c}_i = \sum_{k=1}^{n} \boldsymbol{a}_{ik}\boldsymbol{b}_k, \quad \text{for } i = 1, \ldots, n. \tag{2.6}$$

Similarly, if $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{IR}^n$, the components of $\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b}$ are

$$\boldsymbol{c}_i = \boldsymbol{a}_i + \boldsymbol{b}_i, \quad \text{for } i = 1, \ldots, n. \tag{2.7}$$

2.1.1. Some definitions. If $\underline{\boldsymbol{a}} = \overline{\boldsymbol{a}}$ then $\boldsymbol{a}$ is a *thin* interval; if $\underline{\boldsymbol{a}} = -\overline{\boldsymbol{a}}$ then $\boldsymbol{a}$ is *symmetric*. Two intervals $\boldsymbol{a}$ and $\boldsymbol{b}$ are equal if $\underline{\boldsymbol{a}} = \underline{\boldsymbol{b}}$ and $\overline{\boldsymbol{a}} = \overline{\boldsymbol{b}}$.

We have an inclusion of intervals

$$\boldsymbol{a} \subseteq \boldsymbol{b} \iff \underline{\boldsymbol{a}} \geq \underline{\boldsymbol{b}} \quad \text{and} \quad \overline{\boldsymbol{a}} \leq \overline{\boldsymbol{b}}.$$

Inclusion for interval matrices (and vectors) is defined componentwise by

$$\boldsymbol{A} \subseteq \boldsymbol{B} \iff \boldsymbol{a}_{ij} \subseteq \boldsymbol{b}_{ij} \quad (\text{for all } i, j).$$

For an interval $\boldsymbol{a}$ we define *midpoint*

$$\mathrm{m}(\boldsymbol{a}) = (\underline{\boldsymbol{a}} + \overline{\boldsymbol{a}})/2, \tag{2.8}$$

width

$$\mathrm{w}(\boldsymbol{a}) = \overline{\boldsymbol{a}} - \underline{\boldsymbol{a}}, \tag{2.9}$$

and *magnitude*

$$|\boldsymbol{a}| = \max\bigl\{|\underline{\boldsymbol{a}}|, |\overline{\boldsymbol{a}}|\bigr\}. \tag{2.10}$$

Midpoint, width, and magnitude are defined componentwise for interval vectors and matrices.

The *maximum norm* of an interval vector $\boldsymbol{a} \in \mathbb{IR}^n$ is given by

$$\|\boldsymbol{a}\| = \max_{1\leq i\leq n} |\boldsymbol{a}_i|, \tag{2.11}$$

and of a matrix $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ by

$$\|\boldsymbol{A}\| = \max_{1\le i\le n} \sum_{j=1}^{n} |\boldsymbol{a}_{ij}|\,. \tag{2.12}$$

We also use $\|\cdot\|$ to denote the maximum norm of scalar vectors and matrices.

Let $\mathcal{A}, \mathcal{B} \in \mathbb{R}^n$ be compact non-empty sets. With $q\big(\mathcal{A}, \mathcal{B}\big)$ we denote the Hausdorff distance between $\mathcal{A}$ and B,

$$q\big(\mathcal{A}, \mathcal{B}\big) = \max\{\max_{x\in\mathcal{A}} \min_{y\in\mathcal{B}} \|x-y\|,\ \max_{y\in\mathcal{B}} \min_{x\in\mathcal{A}} \|x-y\|\}. \tag{2.13}$$

By (2.13), the *distance* between two intervals $\boldsymbol{a}$ and $\boldsymbol{b}$ is

$$q(\boldsymbol{a}, \boldsymbol{b}) = \max\{|\underline{\boldsymbol{a}} - \underline{\boldsymbol{b}}|, |\overline{\boldsymbol{a}} - \overline{\boldsymbol{b}}|\}, \tag{2.14}$$

and the distance between two interval vectors $\boldsymbol{u}$ and $\boldsymbol{v} \in \mathbb{IR}^n$ is

$$q(\boldsymbol{u}, \boldsymbol{v}) = \max_{1\le i\le n} q(\boldsymbol{u}_i, \boldsymbol{v}_i). \tag{2.15}$$

2.1.2. Some properties. For real intervals $\boldsymbol{a}$, $\boldsymbol{a}_1$, $\boldsymbol{b}$, and $\boldsymbol{b}_1$ such that $\boldsymbol{a} \subseteq \boldsymbol{a}_1$ and $\boldsymbol{b} \subseteq \boldsymbol{b}_1$, we have

$$\boldsymbol{a} \circ \boldsymbol{b} \subseteq \boldsymbol{a}_1 \circ \boldsymbol{b}_1, \quad \circ \in \{+, -, \cdot, /\}.$$

That is, the interval-arithmetic operations are inclusion monotone.

Although interval addition and multiplication are associative, the distributive law does not hold in general [1]. That is, we can easily find three intervals $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, for which

$$\boldsymbol{a}(\boldsymbol{b} + \boldsymbol{c}) \neq \boldsymbol{a}\boldsymbol{b} + \boldsymbol{a}\boldsymbol{c}.$$

However, for any three intervals $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, the sub-distributive law

$$\boldsymbol{a}(\boldsymbol{b} + \boldsymbol{c}) \subseteq \boldsymbol{a}\boldsymbol{b} + \boldsymbol{a}\boldsymbol{c}, \tag{2.16}$$

does hold. Moreover, there are important cases in which the distributive law

$$\boldsymbol{a}(\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a}\boldsymbol{b} + \boldsymbol{a}\boldsymbol{c}$$

does hold. For example, it holds if $\boldsymbol{b}\boldsymbol{c} \ge 0$, if $\boldsymbol{a}$ is a thin interval, or if $\boldsymbol{b}$ and $\boldsymbol{c}$ are symmetric. In particular, for $\alpha \in \mathbb{R}$, which can be interpreted as the thin interval $[\alpha, \alpha]$, and intervals $\boldsymbol{a}$ and $\boldsymbol{b}$, we have

$$\alpha(\boldsymbol{a} + \boldsymbol{b}) = \alpha\boldsymbol{a} + \alpha\boldsymbol{b}. \tag{2.17}$$

Using (2.2–2.4) and (2.9), one can easily show that

$$\mathrm{w}(\boldsymbol{a} \pm \boldsymbol{b}) = \mathrm{w}(\boldsymbol{a}) + \mathrm{w}(\boldsymbol{b}) \quad \text{and} \tag{2.18}$$

$$\mathrm{w}(\alpha\boldsymbol{a}) = |\alpha|\,\mathrm{w}(\boldsymbol{a}), \tag{2.19}$$

for any $\alpha \in \mathbb{R}$.

Let A be a real matrix. We denote by $|A|$ the matrix with components $|a_{ij}|$, for all i and j. Let $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{IR}^n$, and $\boldsymbol{c} \in \mathbb{IR}^n$. We can easily obtain from (2.6, 2.7, 2.17, 2.18, 2.19) that

$$\mathrm{w}(\boldsymbol{b} \pm \boldsymbol{c}) = \mathrm{w}(\boldsymbol{b}) + \mathrm{w}(\boldsymbol{c}) \quad \text{and} \tag{2.20}$$

$$\mathrm{w}(A\boldsymbol{b}) = |A|\mathrm{w}(\boldsymbol{b}). \tag{2.21}$$

For $A, B \in \mathbb{R}^{n\times n}$ and $\boldsymbol{c} \in \mathbb{IR}^n$, using (2.21),

$$(AB)\boldsymbol{c} \subseteq A(B\boldsymbol{c}). \tag{2.22}$$

For $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{IR}^n$, and $\boldsymbol{c} \in \mathbb{IR}^n$, by (2.20, 2.21),

$$A(\boldsymbol{b} + \boldsymbol{c}) = A\boldsymbol{b} + A\boldsymbol{c}. \tag{2.23}$$

For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{IR}^n$, by (2.10, 2.11, 2.14, 2.15),

$$q(\boldsymbol{a}, \boldsymbol{a} + \boldsymbol{b}) = \|\boldsymbol{b}\|. \tag{2.24}$$

2.2. Ranges of functions. Let $f : \mathbb{R}^m \to \mathbb{R}$ be a function on $\mathcal{D} \subseteq \mathbb{R}^m$. The interval-arithmetic evaluation of f on $\boldsymbol{a} \subseteq \mathcal{D}$, which we denote by $f(\boldsymbol{a})$, is obtained by replacing each occurrence of a real variable with a corresponding interval, by replacing the standard functions with enclosures of their ranges, and by performing interval-arithmetic operations instead of the real operations.

In practice, $f(\boldsymbol{a})$ is not unique, because it depends on how f is evaluated in interval arithmetic. For example, expressions that are mathematically equivalent for scalars, such as $x(y + z)$ and $xy + xz$, may have different values if x, y, and z are intervals; cf. (2.16). However, since we are interested in the interval-arithmetic evaluation of f on a computer, we can assume that $f(\boldsymbol{a})$ is uniquely defined by the code list, or computational graph, of f. No matter how $f(\boldsymbol{a})$ is evaluated, it follows from the inclusion monotone property of the the interval operations that the range of f, $\{f(x) \mid x \in \boldsymbol{a}\}$, is always contained in $f(\boldsymbol{a})$. That is,

$$\big\{f(x) \mid x \in \boldsymbol{a}\big\} \subseteq f(\boldsymbol{a}).$$

2.3. Automatic generation of Taylor coefficients. We require computing Taylor coefficients for the solution to an ODE. Given a computer program for evaluating the right side of the ODE $y' = f(t, y)$, such coefficients can be generated using AD techniques; see for example [3, 8, 15, 23]. It can be shown that to generate k Taylor coefficients, we need at most $O(k^2)$ times as much computational work as we require to evaluate $f(t, y)$ [15]. This is far more efficient than the standard symbolic generation of Taylor coefficients.

In this subsection, we do not present details about the algorithms involved in computing Taylor coefficients—the reader is referred to the above references. We only introduce the following notation, which is used later.

For the IVP $y' = f(t, y)$, $y(t^*) = v$, the ith Taylor coefficient of $y(t; t^*, v)$ at t^*, $(y)_i := y^{(i)}(t^*)/i!$, satisfies

$$(y)_i = \frac{1}{i}\big(f\big)_{i-1}, \quad i \geq 1, \tag{2.25}$$

where $\big(f\big)_{i-1}$ denotes the $(i-1)$st Taylor coefficient of f evaluated at (t^*, v). We shall denote the ith Taylor coefficient of the solution to $y' = f(t, y)$, $y(t^*) = v$ at t^* by $f^{[i]}(t^*, v)$.

If we have a procedure to compute the point Taylor coefficient $f^{[i]}(t^*, v)$ and perform the computations in interval arithmetic with intervals $\boldsymbol{t}^* \ni t^*$ and $\boldsymbol{v} \ni v$, instead of points t^* and v, we obtain a procedure for enclosing this coefficient, since

$$\left\{ f^{[i]}(t^*, v) \mid t^* \in \boldsymbol{t}^*, v \in \boldsymbol{v} \right\} \subseteq f^{[i]}(\boldsymbol{t}^*, \boldsymbol{v}).$$

2.4. Existence and uniqueness of a solution. Since we assume that A and $g \in C^{p-1}(D)$, with $p \geq 1$, by standard ODE theory, for any $t^* \in D$ and any $v \in \mathbb{R}^n$, there exists a unique solution to

$$y' = A(t)y + g(t), \quad y(t^*) = v \tag{2.26}$$

on D.

Hence, we do not have to validate existence and uniqueness of a solution over a set of initial conditions, as in Algorithm I; cf. section 1. In our method, we need to enclose the solution to (2.26) at some $t^* + h$, where h will be a given stepsize.

2.5. Enclosing a point solution. We employ the *high-order enclosure* (HOE) method [21] to compute bounds on the solution of problems in the form (2.26). This method is founded on the following result [6, 21]. For the IVP $y' = f(t, y)$, $y(t^*) = v$, if h and $\widetilde{\boldsymbol{v}}$ are such that

$$v + \sum_{i=1}^{p-1} (t - t^*)^i f^{[i]}(t^*, v) + (t - t^*)^p f^{[p]}(t^*, \widetilde{\boldsymbol{v}}) \subseteq \widetilde{\boldsymbol{v}} \tag{2.27}$$

($p \geq 1$) holds for all $t \in [t^*, t^* + h]$, then this problem has a unique solution $y(t; t^*, v)$ that satisfies $y(t; t^*, v) \in \widetilde{\boldsymbol{v}}$ for all $t \in [t^*, t^* + h]$. Then,

$$y(t^* + h; t^*, v) \in \boldsymbol{v} := v + \sum_{i=1}^{p-1} h^i f^{[i]}(t^*, v) + h^p f^{[p]}(t^*, \widetilde{\boldsymbol{v}}) \subseteq \widetilde{\boldsymbol{v}}. \tag{2.28}$$

An efficient algorithm for computing $\widetilde{\boldsymbol{v}}$ and h, such that (2.27) holds, is given in [21].

3. Constructing an enclosure on the solution. First, we outline our method. Then, we describe the representation of the enclosure on the solution we use and the steps of this method.

3.1. Method outline. The solution to (1.1) with an initial condition v at t_{j-1} is

$$y(t; t_{j-1}, v) = Y(t)v + Y(t) \int_{t_{j-1}}^{t} Y^{-1}(s) g(s) \, \mathrm{d}s, \tag{3.1}$$

where $Y(t)$ is the solution to

$$Y'(t) = A(t)Y(t), \quad Y(t_{j-1}) = I.$$

(I is the $n \times n$ identity matrix.)

Assume that we have computed an enclosure set $\mathcal{S}_{j-1}$, such that

$$y(t_{j-1}; t_0, \boldsymbol{y}_0) \subseteq \mathcal{S}_{j-1}.$$

Then, at the next point t_j,

$$\begin{aligned} y(t_j; t_0, \boldsymbol{y}_0) &\subseteq y(t_j; t_{j-1}, \mathcal{S}_{j-1}) \\ &\subseteq \mathcal{Y}_j := \left\{ Y(t_j)v + Y(t_j) \int_{t_{j-1}}^{t_j} Y(s)^{-1} g(s)\,\mathrm{d}s \mid v \in \mathcal{S}_{j-1} \right\}. \end{aligned} \tag{3.2}$$

We enclose

$$\left\{ Y(t_j)v \mid v \in \mathcal{S}_{j-1} \right\} \quad \text{and} \quad Y(t_j) \int_{t_{j-1}}^{t_j} Y(s)^{-1} g(s)\,\mathrm{d}s,$$

and compute an enclosure set $\mathcal{S}_j$ and an interval vector $\boldsymbol{y}_j$ such that

$$\mathcal{Y}_j \subseteq \mathcal{S}_j \quad \text{and} \tag{3.3}$$
$$\mathcal{Y}_j \subseteq \boldsymbol{y}_j. \tag{3.4}$$

From (3.2–3.4),

$$\begin{aligned} &y(t_j; t_0, \boldsymbol{y}_0) \subseteq \mathcal{S}_j \quad \text{and} \\ &y(t_j; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_j. \end{aligned}$$

We use $\mathcal{S}_j$ as the set of initial conditions for the next integration step, and $\boldsymbol{y}_j$ as an interval-vector enclosure, which can be given to the user as an output.

3.2. Enclosure representation. We represent the enclosure on $y(t_j; t_0, \boldsymbol{y}_0)$ as

$$\mathcal{S}_j = \left\{ u_j + S_j\alpha + B_j r \mid \alpha \in \boldsymbol{\alpha},\, r \in \boldsymbol{r}_j \right\}, \tag{3.5}$$

where $u_j, \alpha, r \in \mathbb{R}^n$, $\boldsymbol{r}_j \in \mathbb{IR}^n$; $S_j, B_j \in \mathbb{R}^{n\times n}$, and B_j is nonsingular; and

$$\boldsymbol{\alpha} = \boldsymbol{y}_0 - \mathrm{m}(\boldsymbol{y}_0).$$

Initially, when $j = 0$,

$$\begin{aligned} u_0 &= \mathrm{m}(\boldsymbol{y}_0), \\ S_0 &= I, \\ B_0 &= I, \quad \text{and} \\ \boldsymbol{r}_0 &= 0. \end{aligned}$$

In (3.5), the set $\left\{ u_j + S_j\alpha \mid \alpha \in \boldsymbol{\alpha} \right\}$ will be an approximation to $y(t_j; t_0, \boldsymbol{y}_0)$, and the set $\{ B_j r \mid r \in \boldsymbol{r}_j \}$ will be an approximation of the overestimation, or excess (considered in section 5) accumulated in the integration process from t_0 to t_j.

Remark 3.1. The representation (3.5) is similar to the representation used in "enclosure method 4" in the AWA program [13].

3.3. Steps of the method. We assume that, at t_{j-1}, we have computed u_{j-1}, S_{j-1}, B_{j-1}, and $\boldsymbol{r}_{j-1}$ such that

$$y(t_{j-1}; t_0, \boldsymbol{y}_0) \subseteq \mathcal{S}_{j-1} = \left\{ u_{j-1} + S_{j-1}\alpha + B_{j-1} r \mid \alpha \in \boldsymbol{\alpha}, r \in \boldsymbol{r}_{j-1} \right\}.$$

Our goal is to find $\mathcal{S}_j$ at t_j. That is, we have to compute u_j, S_j, B_j, and $\boldsymbol{r}_j$.

We show in the remainder of this subsection how we enclose

$$Y(t_j) \int_{t_{j-1}}^{t_j} Y^{-1}(s) g(s)\,\mathrm{d}s \quad \text{and} \quad \left\{ Y(t_j)v \mid v \in \mathcal{S}_{j-1} \right\}$$

and compute u_j, S_j, B_j, and $\boldsymbol{r}_j$.

3.3.1. Enclosing $Y(t_j)\int_{t_{j-1}}^{t_j} Y^{-1}(s)g(s)\,\mathrm{d}s$. The solution to (1.1) with an initial condition the zero vector (denoted by 0 here) at t_{j-1} is

$$y(t_j; t_{j-1}, 0) = Y(t_j)\int_{t_{j-1}}^{t_j} Y^{-1}(s)g(s)\,\mathrm{d}s.$$

Hence, we can apply the HOE method on $y' = A(t)y + g(t)$, $y(t_{j-1}) = 0$ to compute $\boldsymbol{v}_j$ such that

$$y(t_j; t_{j-1}, 0) = Y(t_j)\int_{t_{j-1}}^{t_j} Y^{-1}(s)g(s)\,\mathrm{d}s \in \boldsymbol{v}_j. \tag{3.6}$$

3.3.2. Enclosing $\{Y(t_j)v \mid v \in \mathcal{S}_{j-1}\}$. First, we show how to enclose $Y(t_j)$. Using the HOE method, we integrate (1.1) with initial conditions $y(t_{j-1}) = e^{(i)}$, for all $i = 1, \ldots, n$, where $e^{(i)}$ is the unit vector with component i equal to one. Denoting the computed enclosures at t_j by $\boldsymbol{z}_j^{(i)}$, $i = 1, \ldots, n$, and using (3.1, 3.6), we have

$$\begin{aligned} Y(t_j)e^{(i)} &= y\big(t_j; t_{j-1}, e^{(i)}\big) - y(t_j; t_{j-1}, 0) \\ &\in \boldsymbol{z}_j^{(i)} - \boldsymbol{v}_j. \end{aligned} \tag{3.7}$$

Let $\boldsymbol{Y}_j$ be the interval matrix with ith column $\boldsymbol{z}_j^{(i)} - \boldsymbol{v}_j$. Then

$$Y(t_j) \in \boldsymbol{Y}_j. \tag{3.8}$$

Suppose now that we have computed $\boldsymbol{Y}_j$ such that (3.8) holds. Denote

$$Y_j = \mathrm{m}(\boldsymbol{Y}_j), \tag{3.9}$$
$$S_j = Y_j S_{j-1}, \tag{3.10}$$
$$C_j = Y_j B_{j-1}, \tag{3.11}$$
$$\boldsymbol{E}_j = \boldsymbol{Y}_j - Y_j, \tag{3.12}$$
$$\widetilde{\boldsymbol{y}}_{j-1} = u_{j-1} + S_{j-1}\boldsymbol{\alpha} + B_{j-1}\boldsymbol{r}_{j-1}, \quad \text{and} \tag{3.13}$$
$$\boldsymbol{d}_j = \boldsymbol{E}_j\widetilde{\boldsymbol{y}}_{j-1}. \tag{3.14}$$

By (3.8–3.14), for any $v = u_{j-1} + S_{j-1}\alpha + B_{j-1}r \in \mathcal{S}_{j-1}$,

$$\begin{aligned} Y(t_j)v &= Y_j v + \big(Y(t_j) - Y_j\big)v \\ &= Y_j u_{j-1} + (Y_j S_{j-1})\alpha + (Y_j B_{j-1})r + \big(Y(t_j) - Y_j\big)v \\ &\in \big\{Y_j u_{j-1} + S_j\alpha + C_j r + d \mid \alpha \in \boldsymbol{\alpha},\, r \in \boldsymbol{r}_{j-1},\, d \in \boldsymbol{d}_j\big\}. \end{aligned} \tag{3.15}$$

3.3.3. Enclosing $y(t_j; t_0, \boldsymbol{y}_0)$. Let

$$v_j = \mathrm{m}(\boldsymbol{v}_j), \tag{3.16}$$
$$u_j = v_j + Y_j u_{j-1}, \tag{3.17}$$
$$\boldsymbol{e}_j = \boldsymbol{d}_j + \boldsymbol{v}_j - v_j, \quad \text{and} \tag{3.18}$$
$$\boldsymbol{r}_j = (B_j^{-1}C_j)\boldsymbol{r_{j-1}} + B_j^{-1}\boldsymbol{e}_j, \tag{3.19}$$

where B_j is nonsingular. The choice of B_j is discussed in section 4.

From (3.2, 3.6) and (3.15–3.19),

$$\begin{aligned} y(t_j; t_0, \boldsymbol{y}_0) &\subseteq \{u_j + S_j\alpha + C_j r + e \mid \alpha \in \boldsymbol{\alpha},\, r \in \boldsymbol{r}_{j-1},\, e \in \boldsymbol{e}_j\} \\ &\subseteq u_j + S_j\boldsymbol{\alpha} + C_j\boldsymbol{r}_{j-1} + \boldsymbol{e}_j \qquad (3.20) \\ &=: \boldsymbol{y}_j \quad \text{and} \\ y(t_j; t_0, \boldsymbol{y}_0) &\subseteq \{u_j + S_j\alpha + B_j r \mid \alpha \in r,\, r \in \boldsymbol{r}_j\} \\ &= \mathcal{S}_j. \qquad (3.21) \end{aligned}$$

We compute $\boldsymbol{y}_j$ as an interval-vector enclosure of the solution at t_j, and use u_j, S_j, B_j, and $\boldsymbol{r}_j$ to continue the integration on the next step.

REMARK 3.2. We have suggested using the HOE method for enclosing point solutions. However, any method that can bound a point solution will work in our scheme.

REMARK 3.3. Since, by (3.21),

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq u_j + S_j\boldsymbol{\alpha} + B_j\boldsymbol{r}_j,$$

one may consider intersecting $B_j\boldsymbol{r}_j$ and $C_j\boldsymbol{r}_{j-1} + \boldsymbol{e}_j$ to obtain

$$\widetilde{\boldsymbol{r}}_j = (B_j\boldsymbol{r}_j) \cap (C_j\boldsymbol{r}_{j-1} + \boldsymbol{e}_j),$$

and possibly a tighter enclosure

$$u_j + S_j\boldsymbol{\alpha} + \widetilde{\boldsymbol{r}}_j \subseteq \boldsymbol{y}_j$$

than $\boldsymbol{y}_j$ in (3.20). However, this is unnecessary since $C_j\boldsymbol{r}_{j-1} + \boldsymbol{e}_j \subseteq B_j\boldsymbol{r}_j$. Indeed, by (2.22, 2.23, 3.19),

$$\begin{aligned} C_j\boldsymbol{r}_{j-1} + \boldsymbol{e}_j &= (B_jB_j^{-1}C_j)\boldsymbol{r}_{j-1} + (B_jB_j^{-1})\boldsymbol{e}_j \\ &\subseteq B_j\big((B_j^{-1}C_j)\boldsymbol{r}_{j-1}\big) + B_j(B_j^{-1}\boldsymbol{e}_j) \\ &= B_j\big((B_j^{-1}C_j)\boldsymbol{r}_{j-1} + B_j^{-1}\boldsymbol{e}_j\big) \\ &= B_j\boldsymbol{r}_j. \end{aligned}$$

4. Choice of a transformation matrix. We wish to "absorb" the term $\boldsymbol{e}_j$ in (3.20) and obtain the representation in (3.21). Hence, we have to select B_j such that

$$\{C_j r + e \mid r \in \boldsymbol{r}_{j-1},\, e \in \boldsymbol{e}_j\} \subseteq \{B_j r \mid r \in \boldsymbol{r}_j\},$$

and $\{B_j r \mid r \in \boldsymbol{r}_j\}$ is a "good" enclosure of

$$\{C_j r + e \mid r \in \boldsymbol{r}_{j-1},\, e \in \boldsymbol{e}_j\}. \qquad (4.1)$$

By a good enclosure, we mean that it does not introduce a large overestimation on the enclosed set, and that this enclosure is not difficult to compute.

In the next subsection, we discuss the simplest choice for B_j, $B_j = I$, and illustrate the wrapping effect [15], which often arises as a result of this choice. In subsections 4.2 and 4.3, we present the parallelepiped and QR-factorization methods, respectively, for reducing the wrapping effect. In subsection 4.5, we propose a simple approach for combining them.

4.1. The wrapping effect. We could set $B_j = I$ for all j. Then, by (3.11, 3.19),

$$\boldsymbol{r}_j = C_j \boldsymbol{r}_{j-1} + \boldsymbol{e}_j = Y_j \boldsymbol{r}_{j-1} + \boldsymbol{e}_j. \tag{4.2}$$

If we compute $\boldsymbol{r}_j$ as shown above, we generally obtain unacceptably wide bounds, because of the wrapping effect [13, 15]. We illustrate it with a simple example.

EXAMPLE 4.1. Consider

$$y' = Ay = \begin{pmatrix} 1 & -2 \\ 3 & -4 \end{pmatrix} y. \tag{4.3}$$

Suppose that we integrate this problem with some initial condition and a constant stepsize $h = 0.1$. For simplicity in the exposition, we set $T = Y_j = e^{hA} = e^{0.1A}$, for $j = 1, 2, \ldots$, $\boldsymbol{e}_1 = 10^{-13}([-10, 10], [-1, 1])^T$, and $\boldsymbol{e}_j = ([0, 0], [0, 0])^T$ for $j = 2, 3 \ldots$. Then, (4.2) becomes

$$\begin{aligned} \boldsymbol{r}_1 &= \boldsymbol{e}_1, \\ \boldsymbol{r}_j &= T\boldsymbol{r}_{j-1}, \quad j \geq 2. \end{aligned}$$

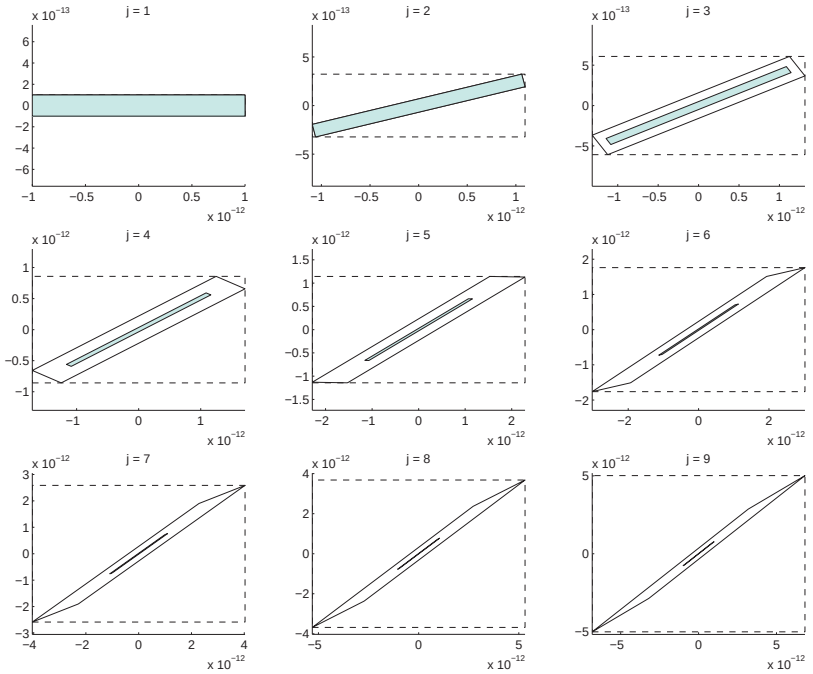
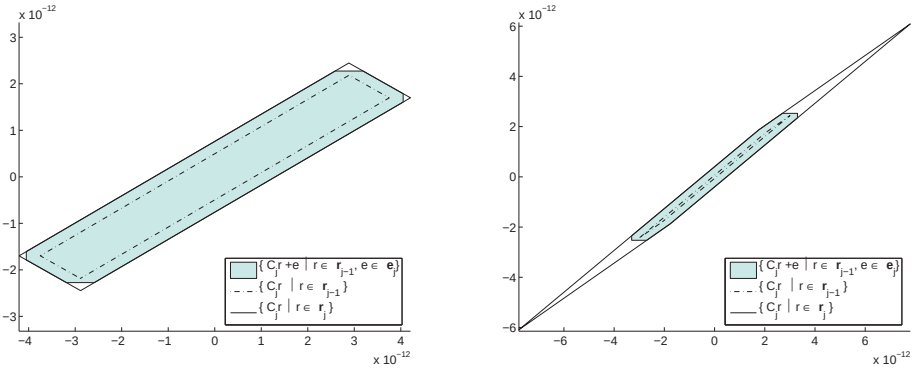


FIG. 4.1. *The sets* $\{T^{j-1}e \mid e \in \boldsymbol{e}_1\}$ *(filled parallelepipeds),* $\{Tr \mid r \in \boldsymbol{r}_{j-1}\}$ *(solid lines), and* $\{r \in \boldsymbol{r}_j\}$ *(dashed lines).*

In Figure 4.1, we show for $j = 1, 2, \ldots, 9$ the boxes specified by $\boldsymbol{r}_j$, the sets $\{Tr \mid r \in \boldsymbol{r}_{j-1}\}$, and the "true" sets $\{T^{j-1}e \mid e \in \boldsymbol{e}_1\}$. As can be seen from this figure, using the interval vector $\boldsymbol{r}_j$ (on each step) as an enclosure of $\{Tr \mid r \in \boldsymbol{r}_{j-1}\}$ introduces an overestimation on this set. Here, $\boldsymbol{r}_j$ "wraps" $\{Tr \mid r \in \boldsymbol{r}_{j-1}\}$. As a result of the wrapping on each step, these overestimations quickly accumulate, leading to the wrapping effect.

4.2. Parallelepiped method. If we select $B_j = C_j$, we obtain the *parallelepiped* method [7, 13]. We shall also refer to it as the P method. In this method, the edges



(a) (b)

FIG. 4.2. *Examples of enclosures produced by the P method.*

of the set $\{C_j r \mid r \in \boldsymbol{r}_j\}$ are parallel to the corresponding edges of $\{C_j r \mid r \in \boldsymbol{r}_{j-1}\}$; see Figure 4.2.

From (3.19),

$$\boldsymbol{r}_j = \boldsymbol{r}_{j-1} + C_j^{-1}\boldsymbol{e}_j.$$

The P method frequently breaks down, because the matrices C_j generally become ill-conditioned, if not singular in floating-point arithmetic. Moreover, this method often leads to large overestimations on the enclosed set [18]; see Figure 4.2(b).

4.3. QR-factorization method. In the QR-factorization method [13], the B_j are orthonormal. Hence, this method cannot break down because of inverting ill-conditioned matrices. Moreover, as shown in [18], the QR-factorization method improves the stability of an interval method.

Here, we consider two versions of this method.

4.3.1. Version 1. We select B_j to be the orthonormal matrix from the QR factorization of C_j. That is,

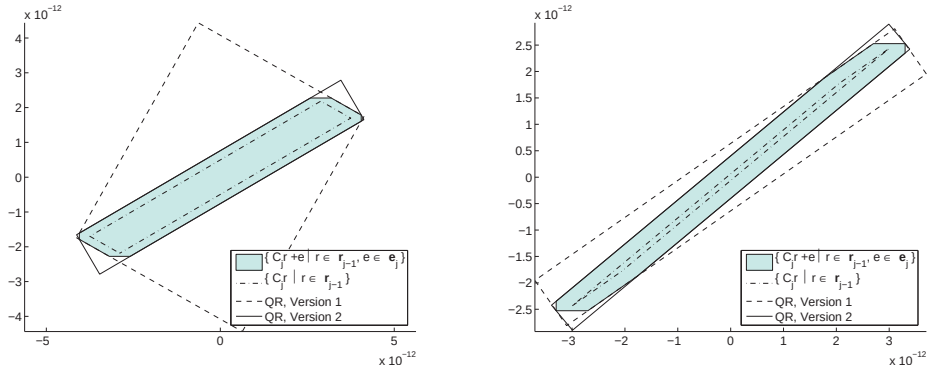
$$C_j = Q_j R_j, \quad B_j = Q_j,$$

where $B_0 = I$. "Geometrically", this approach results in enclosing (4.1) by the set $\{Q_j r \mid r \in \boldsymbol{r}_j\}$ with one of its edges parallel to one of the edges of $\{C_j r \mid r_j \in \boldsymbol{r}_{j-1}\}$; see Figure 4.3. "Algebraically", this method results in an interval method with good stability properties [18].

4.3.2. Version 2. Denote

$$D_{j-1} = \operatorname{diag}\big(\mathrm{w}(\boldsymbol{r}_{j-1})\big).$$

Let P_j be a permutation matrix such that the columns of $C_j D_{j-1} P_j$ are sorted in non-increasing order of their lengths. We perform the QR factorization of $C_j D_{j-1} P_j$



(a) (b)

FIG. 4.3. *Examples of enclosures produced by the QR method. The filled parts are the same as in Figure 4.2.*

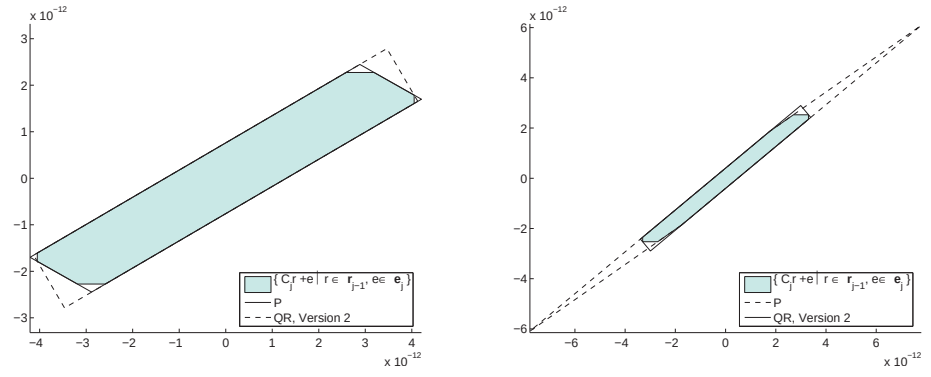
and set B_j to be the orthonormal matrix from this factorization. That is,

$$C_j D_{j-1} P_j = Q_j R_j, \quad B_j = Q_j,$$

where $B_0 = I$. Geometrically, in this version, we enclose (4.1) by $\{Q_j r \mid r \in \boldsymbol{r}_j\}$, whose longest edge is always parallel to the longest edge of (4.1) [13, 18, 20]. Version 2 should generally produce enclosures containing smaller overestimations than the overestimations resulting from Version 1; see Figure 4.3.

In this paper, we use Version 2 and refer to the corresponding method as the *QR* method.

4.4. P versus QR. We "combine" the P method from Figure 4.2 and the QR method from Figure 4.3 to obtain Figure 4.4. Comparing the enclosures in this figure,



(a) (b)

FIG. 4.4. *Enclosures produced by the P and QR methods.*

in part (a), we should select the P method, and in part (b), we should select the QR method.

We show another example illustrating that, on the same problem, the P method can introduce smaller overestimation than the QR method and vice versa.

EXAMPLE 4.2. We assume that we integrate (4.3) with some initial condition and a constant stepsize $h = 0.1$. For simplicity in the exposition, we set for $j = 1, 2, \ldots,$ $T = Y_j = e^{hA} = e^{0.1A}$ and $\boldsymbol{e} = \boldsymbol{e}_j = 10^{-13}([-10, 10], [-1, 1])^T$.

We denote by $\boldsymbol{r}_{\mathrm{QR},j}$ the $\boldsymbol{r}_j$ corresponding to the QR method and by $\boldsymbol{r}_{\mathrm{P},j}$ the $\boldsymbol{r}_j$ from the P method. Similarly, we denote the transformation matrix in the P method by $C_{\mathrm{P},j}$; the transformation matrix in the QR method is denoted by Q_j.

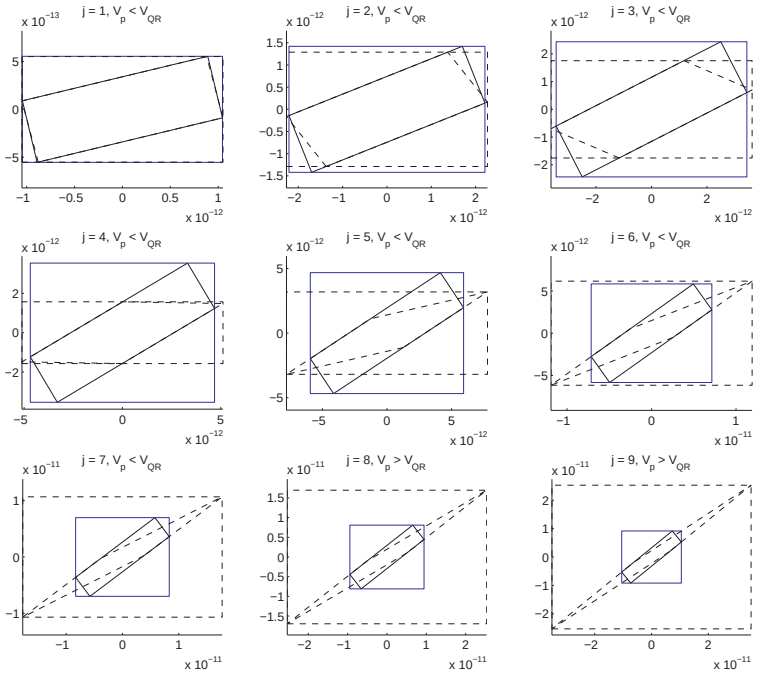


FIG. 4.5. *The sets* $\{C_{\mathrm{P},j}r \mid r \in \boldsymbol{r}_{\mathrm{P},j}\}$ *and* $\{r \in C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j}\}$ *(dashed lines) and* $\{Q_j r \mid r \in \boldsymbol{r}_{\mathrm{QR},j}\}$ *and* $\{r \in Q_j \boldsymbol{r}_{\mathrm{QR},j}\}$ *(solid lines).*

In Figure 4.5, we show the sets

$$\{C_{\mathrm{P},j}r \mid r \in \boldsymbol{r}_{\mathrm{P},j}\} \quad \text{and} \quad \{Q_j r \mid r \in \boldsymbol{r}_{\mathrm{QR},j}\}$$

and the boxes $C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j}$ and $Q_j\boldsymbol{r}_{\mathrm{QR},j}$ enclosing them versus the step number, for $j = 1, \ldots, 9$. We also show the relation between the volumes (denoted by V_{P} and V_{QR}) of these sets. Comparing them and the corresponding boxes, on steps 1 to 5, the P method should compute tighter bounds than the QR method, and on steps 6 to 9, the QR method should compute tighter bounds than the P method.

Although $V_{\mathrm{P}} < V_{\mathrm{QR}}$ on steps 6 and 7, it is reasonable to apply the QR method on these steps, because $Q_j\boldsymbol{r}_{\mathrm{QR},j} \subset C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j}$ for $j = 6, 7$.

4.5. Combining the P and QR methods. Generally, the P method works well, if the condition number of the transformation matrices stay reasonably small. This is normally the case when integrating over relatively short intervals. Over longer

integration intervals, the QR method usually performs better than the P method [18]. Hence, a combination of these two methods is reasonable. In this subsection, we propose a simple scheme for combining them. We refer to the new method as the *QR-P* method.

Suppose that we employ the QR and P methods to compute

$$y(t_{j-1};t_0,\boldsymbol{y}_0) \in \{u_{j-1}+S_{j-1}\alpha+Q_{j-1}r \mid \alpha\in\boldsymbol{\alpha}, r\in\boldsymbol{r}_{\mathrm{QR},j-1}\} \quad \text{and} \tag{4.4}$$
$$y(t_{j-1};t_0,\boldsymbol{y}_0) \in \{u_{j-1}+S_{j-1}\alpha+C_{\mathrm{P},j-1}r \mid \alpha\in\boldsymbol{\alpha}, r\in\boldsymbol{r}_{\mathrm{P},j-1}\}, \tag{4.5}$$

respectively. Following (3.9–3.14) and (3.16–3.18), we denote

$$C_{\mathrm{QR},j} = Y_jQ_{j-1}, \quad Q_0=I, \tag{4.6}$$
$$C_{\mathrm{P},j} = Y_jC_{\mathrm{P},j-1}, \quad C_{\mathrm{P},0}=I, \tag{4.7}$$
$$\boldsymbol{e}_{\mathrm{QR},j} = \boldsymbol{E}_j(u_{j-1}+S_{j-1}\boldsymbol{\alpha}+Q_{j-1}\boldsymbol{r}_{\mathrm{QR},j-1})+\boldsymbol{v}_j-v_j \quad \text{and} \tag{4.8}$$
$$\boldsymbol{e}_{\mathrm{P},j} = \boldsymbol{E}_j(u_{j-1}+S_{j-1}\boldsymbol{\alpha}+C_{j-1}\boldsymbol{r}_{\mathrm{P},j-1})+\boldsymbol{v}_j-v_j. \tag{4.9}$$

In the step from t_{j-1} to t_j, we propagate (4.4) and (4.5) and obtain

$$\begin{aligned} y(t_j;t_0,\boldsymbol{y}_0) &\in \{u_j+S_j\alpha+C_{\mathrm{QR},j}r+e \mid \alpha\in\boldsymbol{\alpha}, r\in\boldsymbol{r}_{\mathrm{QR},j-1}, e\in\boldsymbol{e}_{\mathrm{QR},j}\}\\ &\subseteq u_j+S_j\boldsymbol{\alpha}+C_{\mathrm{QR},j}\boldsymbol{r}_{\mathrm{QR},j-1}+\boldsymbol{e}_{\mathrm{QR},j}\\ &=: \boldsymbol{y}_{\mathrm{QR},j} \end{aligned} \tag{4.10}$$

and

$$\begin{aligned} y(t_j;t_0,\boldsymbol{y}_0) &\in \{u_j+S_j\alpha+C_{\mathrm{P},j}r+e \mid \alpha\in\boldsymbol{\alpha}, r\in\boldsymbol{r}_{\mathrm{P},j-1}, e\in\boldsymbol{e}_{\mathrm{P},j}\}\\ &\subseteq u_j+S_j\boldsymbol{\alpha}+C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j-1}+\boldsymbol{e}_{\mathrm{P},j}\\ &=: \boldsymbol{y}_{\mathrm{P},j} \end{aligned} \tag{4.11}$$

respectively; cf. (3.20).

Let

$$\boldsymbol{p}_j = C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j-1}+\boldsymbol{e}_{\mathrm{P},j}, \tag{4.12}$$
$$\boldsymbol{q}_j = C_{\mathrm{QR},j}\boldsymbol{r}_{\mathrm{QR},j-1}+\boldsymbol{e}_{\mathrm{QR},j}, \tag{4.13}$$
$$\boldsymbol{s}_j = \boldsymbol{p}_j\cap\boldsymbol{q}_j, \quad \text{and} \tag{4.14}$$
$$\boldsymbol{y}_{\text{QR-P},j} = u_j+S_j\boldsymbol{\alpha}+\boldsymbol{s}_j. \tag{4.15}$$

(In (4.14), the intersection is componentwise.) Obviously,

$$\boldsymbol{y}_{\text{QR-P},j}\subseteq\boldsymbol{y}_{\mathrm{QR},j} \quad \text{and} \quad \boldsymbol{y}_{\text{QR-P},j}\subseteq\boldsymbol{y}_{\mathrm{P},j}.$$

Hence, we can compute an interval-vector enclosure that is not worse, and can be better than the enclosures produced by the QR and P methods.

Applying the QR method to (4.10), we have

$$\boldsymbol{r}_{\mathrm{QR},j} = (Q_j^{-1}C_{\mathrm{QR},j})\boldsymbol{r}_{\mathrm{QR},j-1}+Q_j^{-1}\boldsymbol{e}_{\mathrm{QR},j} \quad \text{and}$$
$$y(t_j;t_0,\boldsymbol{y}_0) \in \{u_j+S_j\alpha+Q_jr \mid \alpha\in\boldsymbol{\alpha}, r\in\boldsymbol{r}_{\mathrm{QR},j}\}.$$

Here, Q_j is the orthonormal factor from the QR-factorization of $C_{\mathrm{QR},j}D_{j-1}P_j$, where $D_{j-1}=\mathrm{diag}\big(\mathrm{w}(\boldsymbol{r}_{\mathrm{QR},j-1})\big)$, and P_j is a permutation matrix such that the columns of $C_{\mathrm{QR},j}D_{j-1}P_j$ are sorted in non-increasing order of their lengths.

Similarly, applying the P method to (4.11), we have

$$\boldsymbol{r}_{\mathrm{P},j} = \boldsymbol{r}_{\mathrm{P},j-1} + C_{\mathrm{P},j}^{-1}\boldsymbol{e}_{\mathrm{P},j} \quad \text{and} \tag{4.16}$$

$$y(t_j; t_0, \boldsymbol{y}_0) \in \{u_j + S_j\alpha + C_{\mathrm{P},j}r \mid \alpha \in \boldsymbol{\alpha}, r \in \boldsymbol{r}_{\mathrm{P},j}\}. \tag{4.17}$$

Therefore, we can compute enclosures with both the P and QR methods and intersect $\boldsymbol{p}_j$ and $\boldsymbol{q}_j$ to obtain tighter bounds on the solution, than the bounds produced by the P and QR methods, if executed alone. However, the $\boldsymbol{r}_{\mathrm{P},j}$ may become too wide, and this intersection would not contribute to a better method. Moreover, the P method may break down, because of enclosing a nearly singular, or singular matrix in floating-point arithmetic. To avoid these disadvantages of the P method, we can restart it with $C_{\mathrm{P},j} = Q_j$ and $\boldsymbol{r}_{\mathrm{QR},j} = \boldsymbol{r}_{\mathrm{P},j}$, when this method starts computing "worse" bounds than the QR method. We employ the following simple heuristic for resetting the P method:

if $Q_j\boldsymbol{r}_{\mathrm{QR},j} \subseteq C_{\mathrm{P},j}\boldsymbol{r}_{\mathrm{P},j}$ then

set $C_{\mathrm{P},j} = Q_j$ and $\boldsymbol{r}_{\mathrm{QR},j} = r_{\mathrm{P},j}$.

In Figure 4.6, we show $\{Q_j r \mid r \in \boldsymbol{r}_{\mathrm{QR},j}\}$ and $\{C_{\mathrm{P},j}r \mid r \in \boldsymbol{r}_{\mathrm{P},j}\}$, with the above resetting scheme, on the problem from Example 4.2. Notice that now the P method

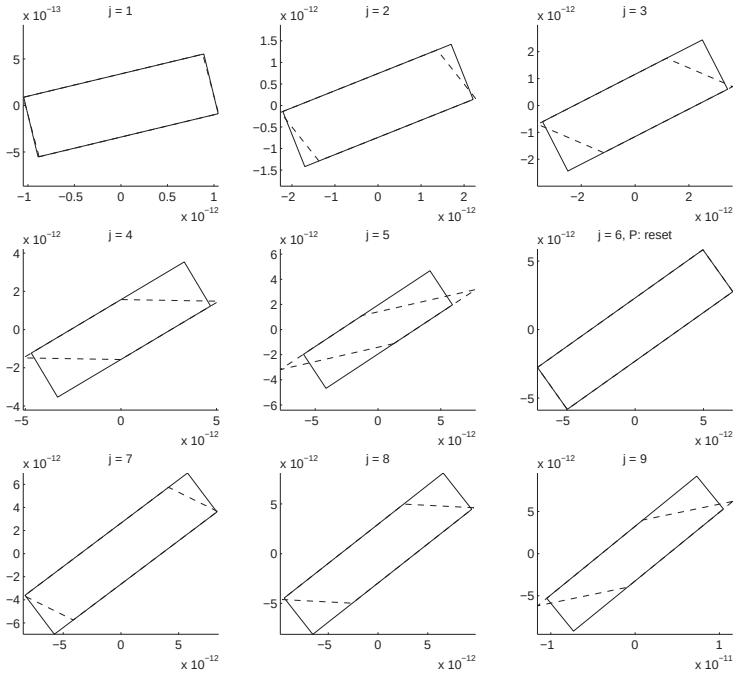


FIG. 4.6. *The sets* $\{C_{\mathrm{P},j}r \mid r \in \boldsymbol{r}_{\mathrm{P},j}\}$ *(dashed lines) and* $\{Q_j r \mid r \in \boldsymbol{r}_{\mathrm{QR},j}\}$ *(solid lines) on the problem from Example 4.2.*

with the resetting computes smaller sets than the QR method; cf. Figure 4.5. In Figure 4.7, we depict the boxes $\boldsymbol{p}_j$, $\boldsymbol{q}_j$, and $\boldsymbol{s}_j$.

5. Global and local excess. We refer to the overestimation in the computed bounds as *excess* [16]. We define the global excess in $\boldsymbol{y}_j \supseteq y(t_j; t_0, \boldsymbol{y}_0)$ by

$$\gamma_j = q\big(y(t_j; t_0, \boldsymbol{y}_0), \boldsymbol{y}_j\big),$$

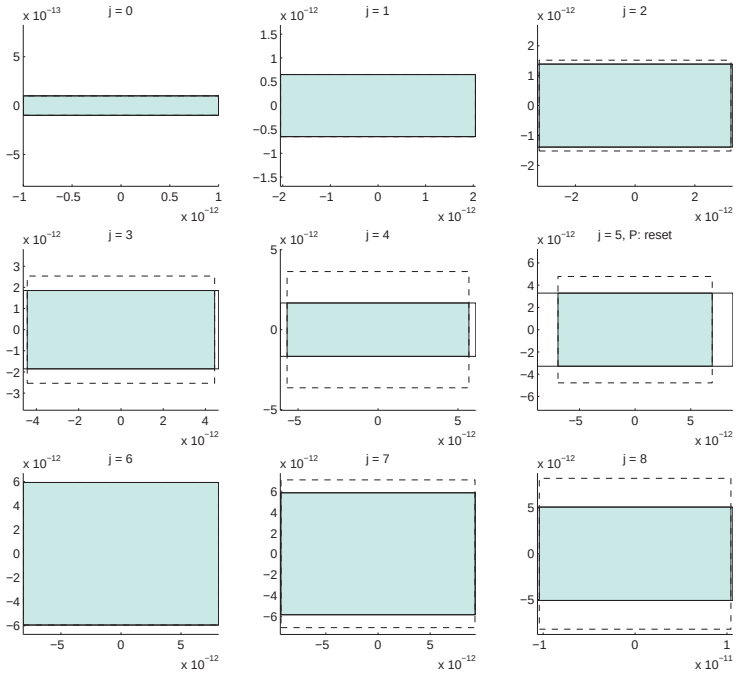


FIG. 4.7. *The boxes $\boldsymbol{p}_j$, solid lines, $\boldsymbol{q}_j$ (dashed lines) and $\boldsymbol{s}_j$ (filled boxes) on the problem from Example 4.2.*

and the local excess by

$$\delta_j = q\big(y(t_j; t_{j-1}, \mathcal{S}_{j-1}), \boldsymbol{y}_j\big),$$

for $j \geq 1$. Similar to standard numerical methods for IVPs for ODEs, it is meaningful to control in an interval method the local excess per step or unit step. Since we can also estimate the global excess, we can report it to the user, to give an indication about the quality of the computed enclosures.

REMARK 5.1. The QR-P method combines the advantages of the P and QR methods. As will be demonstrated in section 7, the QR-P method can compute much tighter enclosures than the QR method at little extra cost.

The blunting approach of Makino and Berz [14] also combines these two methods. In their method, the transformation matrix is chosen to be a sum of the matrix related to the P method and an orthonormal matrix multiplied by a "blunting" factor, which is typically a small number, for example 10^{-3}. In this paper, we do not investigate the blunting approach.

REMARK 5.2. We have proposed the QR-P method for reducing the wrapping effect when integrating linear ODEs. For nonlinear ODEs, the author believes that this method should generally produce tighter enclosures than the QR method. Studies in the nonlinear case, however, are not within the scope of this paper.

5.1. Estimating the global excess. We consider $\{u_j + S_j\alpha \mid \alpha \in \boldsymbol{\alpha}\}$ as an approximation to $y(t_j; t_0, \boldsymbol{y}_0)$. If $y(t_j; t_0, \boldsymbol{y}_0) \in u_j + S_j\boldsymbol{\alpha} + \boldsymbol{z}_j$, we approximate the global excess by

$$\begin{aligned} \mathrm{ge}_j &= q\big(u_j + S_j\boldsymbol{\alpha}, \boldsymbol{y}_j\big) \\ &= q\big(u_j + S_j\boldsymbol{\alpha}, u_j + S_j\boldsymbol{\alpha} + \boldsymbol{z}_j\big), \\ &= \|\boldsymbol{z}_j\|, \end{aligned} \tag{5.1}$$

where the third equality follows from (2.24). We have $\boldsymbol{z}_j = \boldsymbol{p}_j$, $\boldsymbol{q}_j$, $\boldsymbol{s}_j$ in the P, QR, and QR-P methods, respectively; cf. (4.12, 4.13, 4.14).

5.2. Estimating the local excess. From (3.14, 3.18),

$$\boldsymbol{e}_j = \boldsymbol{E}_j\widetilde{\boldsymbol{y}}_{j-1} + (\boldsymbol{v}_j - v_j).$$

The interval matrix $\boldsymbol{E}_j$ is an enclosure of the overestimation when bounding the solution to $Y' = A(t)Y$, $Y(t_{j-1}) = I$ at t_j; $\boldsymbol{v}_j - v_j$ is an enclosure of the excess when integrating (1.1) with $y(t_{j-1}) = 0$. Hence, we can view $\boldsymbol{e}_j$ as an enclosure on an approximation of the local excess.

In the QR-P and QR methods, we estimate it by

$$\mathrm{le}_j = \|\boldsymbol{e}_{\mathrm{QR},j}\|; \tag{5.2}$$

cf. (4.8) and (3.18).

6. Stepsize control. Denote by Atol and Rtol absolute and relative tolerances (supplied by the user), respectively, and set

$$\mathrm{Tol}_j = \mathrm{Rtol} \cdot \|\boldsymbol{y}_j\| + \mathrm{Atol}.$$

We adopt local excess control per unit step (LEPUS). Namely, we require on each step that

$$\|\mathrm{le}_j\| \le h_j \mathrm{Tol}_j. \tag{6.1}$$

The local excess estimation le_j behaves like $c_j h_j^p$, where $c_j > 0$ is a real constant, and p is the order of the Taylor series.

Initial Stepsize. On the first attempt on the first step, we compute [21]

$$h_0 = \left(\frac{\mathrm{Tol}_0}{\|(p+1)f^{[p+1]}(\boldsymbol{y}_0)\|}\right)^{1/p}.$$

Successful step. Let

$$\theta_j = 0.9\left(\frac{0.5h_j\mathrm{Tol}_j}{\mathrm{le}_j}\right)^{\frac{1}{p-1}} \tag{6.2}$$

and

$$\beta_j = \max\big(0.5, \min(2.0, \theta_j)\big). \tag{6.3}$$

If (6.1) holds, we select

$$h_{j+1} = \beta_j h_j;$$

see for example [5]. In (6.2), 0.5 and 0.9 are safety factors. In (6.3), we ensure that the selected stepsize is at most twice the last stepsize and at least half the last stepsize.

Unsuccessful step. If (6.1) does not hold, we reduce h_j using

$$\widehat{h}_j = h_j \left(\frac{h_j \mathrm{Tol}_j}{\mathrm{le}_j} \right)^{\frac{1}{p-1}} . \tag{6.4}$$

Assuming that we store each Taylor coefficient multiplied by the corresponding power of the stepsize, when a step is rejected, we only need to rescale this coefficient.

This scheme for stepsize control has shown to work well. In our experiments, usually one or two stepsize rejections would occur per integration, normally at its beginning.

7. Numerical results. We study empirically the QR-P method and compare it with the QR method. The numerical results in this section are produced with a linear solver implementing the methods presented in this paper.[1] In our computations, we have used constant order 17 and variable stepsize control, as described in section 6. The platform is a Sun UltraSparc 10 with 450MHz CPU and 512 RAM. The compiler used is GNU C++ version 3.2.

In the tables below we present the following quantities.

Tol	Value for Atol and Rtol, where we set Atol = Rtol = Tol. We use Tol = $10^{-7}, 10^{-9}, 10^{-11}$, and 10^{-13}.
GE	estimated global excess at t_{end}; cf. (5.1). The columns with a heading QR-P contain global excess corresponding to the QR-P method, and the columns with a heading R contain ratios of the excess in the QR-P method to the corresponding excess in the QR method.
Time	user CPU time in seconds. The columns with a heading QR-P contain times corresponding to the QR-P method, and the columns with a heading R contain ratios of the time taken by the QR-P method to the time taken by the QR method.
Steps	number of steps. The numbers after "/" are the number of resets in the P method; cf. subsection 4.5.

We consider two problems of the form $y' = Ay$, where A is a constant matrix, a problem of the form $y' = A(t)y$, and a problem of the form $y' = A(t)y + g(t)$. The numerical results on these problems capture the essence in the performance of the QR-P and QR methods.

7.1. Constant coefficient problems. We have not observed a substantial difference in the global excess of the QR and QR-P methods when applied to constant coefficient problems. This can be explained with the theory developed in [18], where it is shown that the global excess in the QR method cannot be much larger than the corresponding global error in a point Taylor series method. However, the QR-P method is generally more expensive than the QR method, because of the extra work in the P part of the QR-P method.

Here, we present numerical results, to give the reader an indication about the performance of these two methods.

[1] This solver will be part of the VNODE package [19].

PROBLEM 7.1.

$$y' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} y \qquad (7.1)$$
$$y(0) = ([1, 11], [10, 11])^T, \quad t_{\text{end}} = 1000.$$

The eigenvalues are $\pm i$. Since the eigenvalues of Y_j are of nearly the same magnitude, the QR and P methods should produce nearly the same global excess [18]. Therefore, the QR and QR-P method should produce almost the same global excess. This is confirmed in Figure 7.1, where we plot ge_j versus t_j when $\text{Tol} = 10^{-9}$. In Figure 7.1, the global excess in these methods is indistinguishable when plotted.

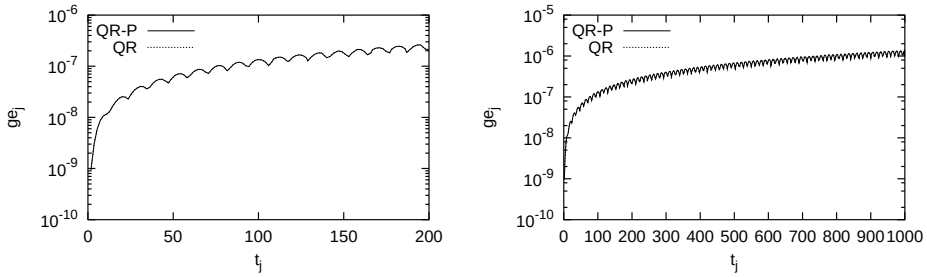


FIG. 7.1. *Global excess versus time in the QR and QR-P methods on (7.1);* $\text{Tol} = 10^{-9}$.

From Table 7.1, the QR-P method is about 6% to 12% more expensive than the QR method. (These numbers are rough approximations, as timing results are generally not very accurate.)

TABLE 7.1
The QR-P and QR methods on (7.1)

Tol	GE		Time		Steps	
	QR-P	R	QR-P	R	QR-P	QR
10^{-7}	1.0×10^{-4}	1.0×10^{0}	0.4	0.87	431/4	431
10^{-9}	1.3×10^{-6}	1.0×10^{0}	0.8	1.12	553/5	553
10^{-11}	1.7×10^{-8}	1.0×10^{0}	0.8	1.13	710/7	710
10^{-13}	2.1×10^{-10}	1.0×10^{0}	1.0	1.06	914/7	914

PROBLEM 7.2.

$$y' = \begin{pmatrix} 1 & -2 \\ 3 & -4 \end{pmatrix} y \qquad (7.2)$$
$$y(0) = ([0.5, 5.5], [-1, 0])^T, \quad t_{\text{end}} = 1000.$$

The eigenvalues are -1 and -2. As in the previous example, when plotted, the global excess in the QR method is indistinguishable from the global excess in the QR-P method; see Figure 7.2.

When the eigenvalues are of distinct magnitudes, over sufficiently long integration intervals, the P method generally produces large overestimations and breaks down, because the transformation matrices become ill conditioned; cf. [18]. Here, the P method is reset nearly on every step, Table 7.2.

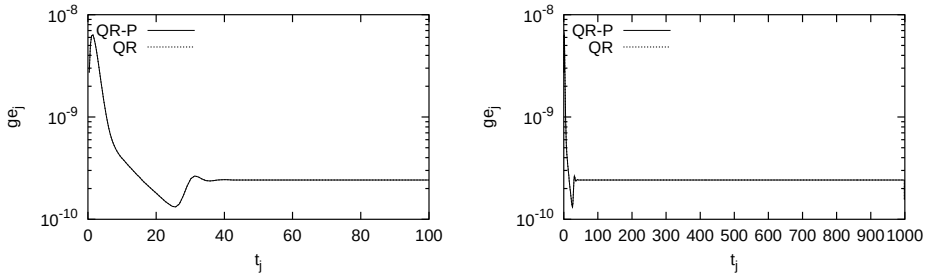


FIG. 7.2. *Global excess versus time in the QR and QR-P methods on (7.2);* Tol $= 10^{-9}$.

TABLE 7.2
The QR-P and QR methods on (7.2)

Tol	GE		Time		Steps	
	QR-P	R	QR-P	R	QR-P	QR
10^{-7}	1.6×10^{-8}	1.0×10^{0}	1.4	1.14	914/911	914
10^{-9}	1.5×10^{-10}	1.0×10^{0}	1.3	0.97	924/914	924
10^{-11}	1.5×10^{-12}	1.0×10^{0}	1.3	1.08	938/920	938
10^{-13}	1.1×10^{-14}	1.0×10^{0}	1.4	1.08	957/924	957

7.2. General linear problems.

PROBLEM 7.3.

$$y' = \begin{pmatrix} 0 & 1 \\ -t^2 & 0 \end{pmatrix} y, \quad y(0) \in \begin{pmatrix} [0.9, 1.1] \\ [-1.1, -0.9] \end{pmatrix}, \tag{7.3}$$
$$t_{\text{end}} = 200.$$

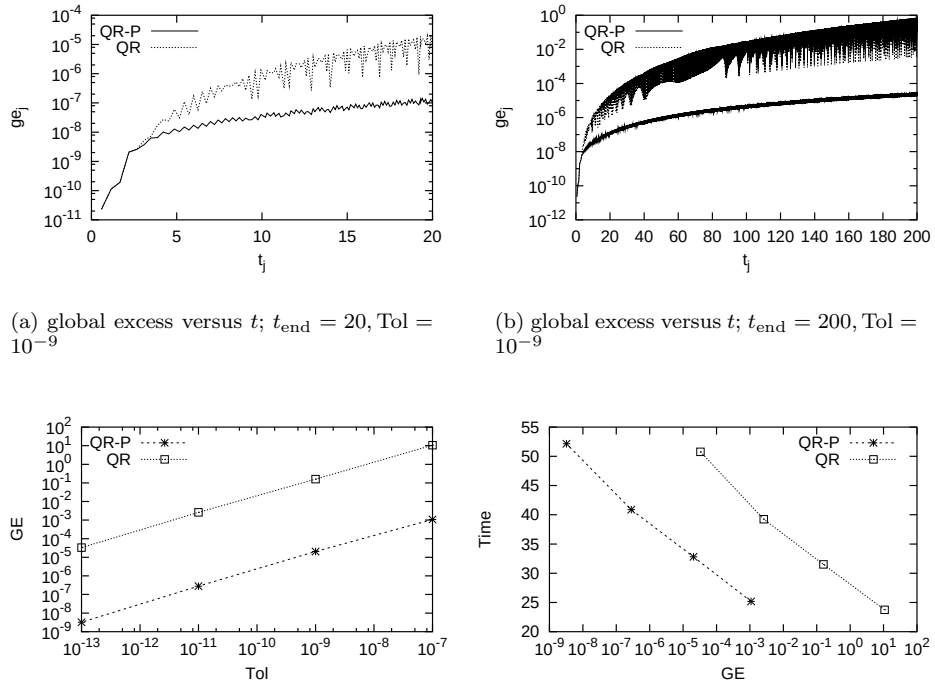
On this problem, for approximately a 3% to 6% increase in the computing time, the QR-P method produces significantly smaller global excess than that produced by the QR method; see Table 7.3 and Figure 7.3.

TABLE 7.3
The QR-P and QR methods on (7.3)

Tol	GE		Time		Steps	
	QR-P	R	QR-P	R	QR-P	QR
10^{-7}	1.1×10^{-3}	1.0×10^{-4}	25.2	1.06	8942/3	8708
10^{-9}	2.0×10^{-5}	1.3×10^{-4}	32.8	1.04	11179/3	11170
10^{-11}	2.8×10^{-7}	1.1×10^{-4}	40.9	1.04	14343/2	14343
10^{-13}	3.2×10^{-9}	9.8×10^{-5}	52.2	1.03	18575/2	18575

The plots in Figure 7.3(a) and (b) illustrate the behavior of the global excess versus t in these two methods when Tol $= 10^{-9}$. The plot in Figure 7.3(c) shows good proportionality of the global excess at t_{end} with respect to the tolerance. Such a proportionality is desirable in numerical methods for IVPs for ODEs [28, 29]. Finally, the better efficiency of the QR-P method is clearly seen in Figure 7.3(d).

On this and the next problem, for the same tolerance, the QR-P method is not much more expensive that the QR method. This is explained as follows. As the right hand side of the ODE becomes more complicated in terms of arithmetic operations and elementary functions, the work that goes into computing Taylor coefficients increases,



(a) global excess versus t; $t_{\text{end}} = 20$, Tol $= 10^{-9}$

(b) global excess versus t; $t_{\text{end}} = 200$, Tol $= 10^{-9}$

(c) global excess at t_{end} versus tolerance

(d) CPU time versus global excess at t_{end}

FIG. 7.3. *The QR-P and QR methods on (7.3).*

and the relative work in the P part of the QR-P method with respect to the overall work decreases. Hence, the ratio of the CPU time in the QR-P to the CPU time in the QR method should be close to one.

PROBLEM 7.4.

$$y' = \begin{pmatrix} \sin(t+10) & -2 & -1 \\ 3 & -4\cos(t^2) & 0 \\ e^{-t^2} & -e^{-t^2} & 0 \end{pmatrix} y + \begin{pmatrix} \sin(t) \\ \cos(t) \\ \sin(t) \end{pmatrix}, \quad y(0) \in \begin{pmatrix} [0,5] \\ [-2,6] \\ [5,12] \end{pmatrix}, \tag{7.4}$$

$t_{\text{end}} = 20$.

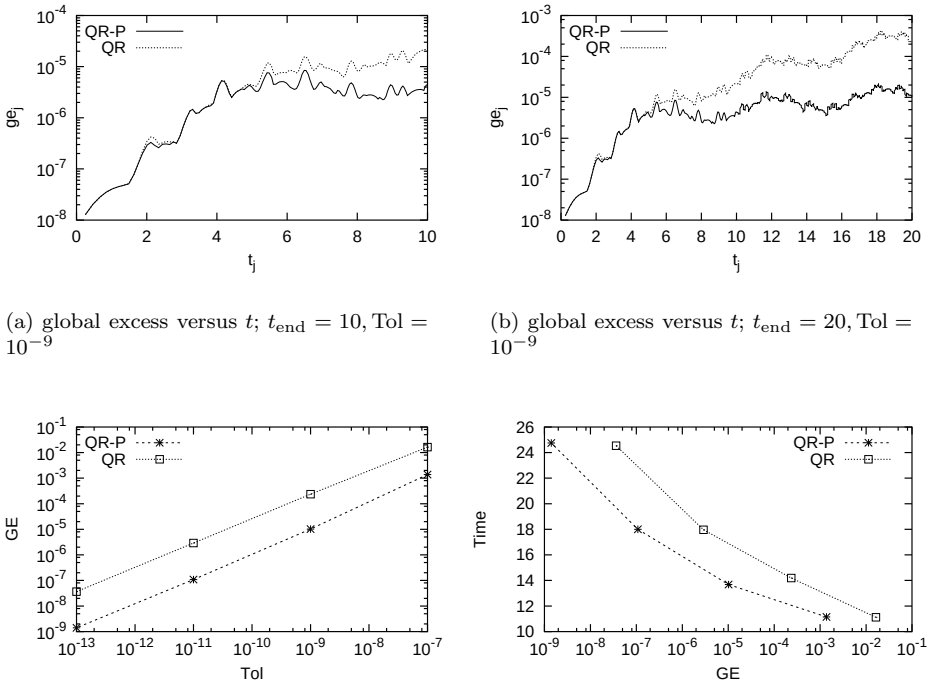
Here, for nearly the same computing time, we achieve about two orders of magnitude improvement in the global excess in the QR-P method, compared to the global excess of the QR method; see Table 7.4 and Figure 7.4.

8. An alternative approach: following the vertices of a parallelepiped. The following idea appears in [12, 22], and it is also discussed in [20, 24].

The solution set $y(t; t_0, \boldsymbol{y}_0)$ is a parallelepiped at any t. An n-dimensional parallelepiped can be determined by $(n+1)$ points. Suppose that $(n+1)$ points specifying a parallelepiped that contains $y(t_{j-1}; t_0, \boldsymbol{y}_0)$ are given. The idea is first to enclose $(n+1)$ point solutions at t_j, originating from the point initial conditions at t_{j-1}, and then to determine $(n+1)$ points that specify a parallelepiped containing $y(t_j; t_0, \boldsymbol{y}_0)$.

TABLE 7.4
The QR-P and QR methods on (7.4)

Tol	GE		Time		Steps	
	QR-P	R	QR-P	R	QR-P	QR
10^{-7}	1.4×10^{-3}	8.4×10^{-2}	11.2	1.00	374/4	374
10^{-9}	1.0×10^{-5}	4.3×10^{-2}	13.7	0.96	474/2	474
10^{-11}	1.1×10^{-7}	3.7×10^{-2}	18.0	1.00	608/3	608
10^{-13}	1.4×10^{-9}	3.9×10^{-2}	24.8	1.01	839/3	839



(a) global excess versus t; $t_{\text{end}} = 10, \text{Tol} = 10^{-9}$

(b) global excess versus t; $t_{\text{end}} = 20, \text{Tol} = 10^{-9}$

(c) global excess at t_{end} versus tolerance

(d) CPU time versus global excess at t_{end}

FIG. 7.4. *The QR-P and QR methods on (7.4).*

To the author's knowledge, this idea has not been pursued in the interval literature. In this section, we present a possible realization of it.

We start with an illustration. When $n = 2$, we can compute bounds on three point solutions; see Figure 8.1(a). These bounds are two-dimensional interval vectors, or boxes. Each of these boxes contains a vertex of the parallelepiped enclosing a true solution. From the computed enclosures, we find a box that contains the fourth vertex; see Figure 8.1(b). Then, we construct a parallelepiped containing these four boxes. Figure 8.1, (c) and (d), show two such parallelepipeds.

Denote by $\mathbf{1}$ the n-dimensional interval vector with each component $[0, 1]$. In general, assume that

$$y(t_{j-1}; t_0, \boldsymbol{y}_0) \subseteq \mathcal{U}_{j-1} := \{b_{j-1} + B_{j-1}\alpha \mid \alpha \in \mathbf{1}\},$$

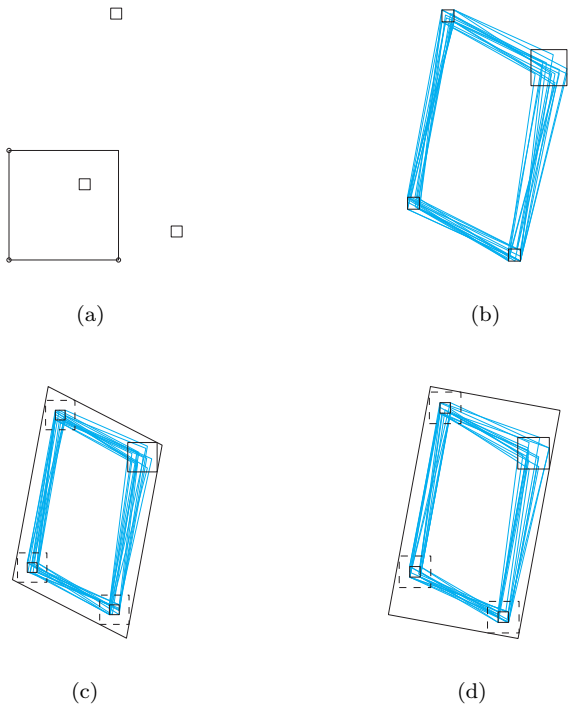


FIG. 8.1. *(a) Initial set at t_0 and enclosures of point solutions at t_1—we integrate with the initial conditions denoted by $\circ$ and obtain enclosures of point solutions at t_1, the three boxes. (b) Some of the parallelepipeds with vertices in these boxes; the larger box contains the fourth vertex. (c), (d) Parallelepipeds enclosing the true solution.*

where $b_{j-1} \in \mathbb{R}^n$, and $B_{j-1} \in \mathbb{R}^{n\times n}$. We wish to compute b_j and B_j at t_j such that

$$y(t_j; t_0, \boldsymbol{y}_0) \subseteq \mathcal{U}_j := \{b_j + B_j\alpha \mid \alpha \in \mathbf{1}\}.$$

Our method consists of two main steps, which are described briefly in the next two paragraphs.

Enclosing point solutions. Denote the ith column of B_{j-1} by $b_{j-1}^{(i)}$ and set

$$v_{j-1}^{(0)} = b_{j-1} \quad \text{and} \quad v_{j-1}^{(i)} = b_{j-1} + b_{j-1}^{(i)}, \quad \text{for } i = 1, \ldots, n.$$

For each $v_{j-1}^{(i)}$, we find $\boldsymbol{w}_j^{(i)}$ such that

$$y\big(t_j; t_{j-1}, v_{j-1}^{(i)}\big) \in \boldsymbol{w}_j^{(i)}.$$

To compute the $\boldsymbol{w}_j^{(i)}$, one can use the HOE method, for example.

Computing a parallelepiped. Having these $(n+1)$ boxes, we determine $b_j \in \mathbb{R}^n$ and $B_j \in \mathbb{R}^{n\times n}$ such that

$$\left\{ Y(t_j)v + Y(t_j)\int_{t_{j-1}}^{t_j} Y(s)^{-1}g(s)\,\mathrm{d}s \mid v \in \mathcal{U}_{j-1} \right\} \subseteq \mathcal{U}_j = \{b_j + B_j\alpha \mid \alpha \in \mathbf{1}\}.$$

The next two subsections describe how b_j and B_j can be computed.

8.1. Computing a parallelepiped. Denote

$$c_j^{(i)} = \mathrm{m}\big(\boldsymbol{w}_j^{(i)}\big), \quad i = 0, \ldots, n, \tag{8.1}$$

$$C_j \quad \text{the } n \times n \text{ matrix with } i\text{th column } c_j^{(i)} - c_j^{(0)}, \tag{8.2}$$

$$\boldsymbol{e}_j^{(i)} = \boldsymbol{w}_j^{(i)} - c_j^{(i)}, \quad i = 0, \ldots, n, \quad \text{and} \tag{8.3}$$

$$\boldsymbol{e}_j = \sum_{i=1}^{n} \boldsymbol{e}_j^{(i)} + (n-1)\boldsymbol{e}_j^{(0)}. \tag{8.4}$$

Proposition 8.1.
For all $v \in \big\{b_{j-1} + B_{j-1}\alpha \mid \alpha \in \mathbf{1}\big\}$,

$$y(t_j; t_{j-1}, v) \in \big\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1}, e \in \boldsymbol{e}_j\big\}. \tag{8.5}$$

Proof. Since $\big|1 - \sum_{i=1}^{n} \alpha_i\big| \le n-1$, where $\alpha_i \in [0,1]$, and $\boldsymbol{e}_j^{(0)}$ is symmetric, for all $e \in \boldsymbol{e}_j^{(0)}$,

$$\Big(1 - \sum_{i=1}^{n} \alpha_i\Big) e \in (n-1)\boldsymbol{e}_j^{(0)}. \tag{8.6}$$

Using (8.1–8.6), for all $\alpha \in \mathbf{1}$, all $w_j^{(i)} \in \boldsymbol{w}_j^{(i)}$, and all $e_j^{(i)} = w_j^{(i)} - c_j^{(i)} \in \boldsymbol{e}_j^{(i)}$,

$$\begin{aligned}
&w_j^{(0)} + \sum_{i=1}^{n} \alpha_i \left(w_j^{(i)} - w_j^{(0)}\right) \\
&\quad = c_j^{(0)} + C_j\alpha + \left(w_j^{(0)} - c_j^{(0)}\right) + \sum_{i=1}^{n} \alpha_i \left(w_j^{(i)} - w_j^{(0)} - \big(c_j^{(i)} - c_j^{(0)}\big)\right) \\
&\quad = c_j^{(0)} + C_j\alpha + \sum_{i=1}^{n} \alpha_i e_j^{(i)} + \Big(1 - \sum_{i=1}^{n} \alpha_i\Big) e_j^{(0)} \\
&\quad \in \big\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1}, e \in \boldsymbol{e}_j\big\}.
\end{aligned}$$

Then, for all $v \in \big\{b_{j-1} + B_{j-1}\alpha \mid \alpha \in \mathbf{1}\big\}$,

$$\begin{aligned}
y(t_j; t_{j-1}, v) &\in \left\{w_j^{(0)} + \sum_{i=1}^{n} \alpha_i\big(w_j^{(i)} - w_j^{(0)}\big) \mid \alpha_i \in \mathbf{1},\, w_j^{(i)} \in \boldsymbol{w}_j^{(i)}\right\} \\
&\subseteq \big\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1}, e \in \boldsymbol{e}_j\big\}.
\end{aligned}$$

□

Now, we have to determine $b_j \in \mathbb{R}^n$ and $B_j \in \mathbb{R}^{n\times n}$ such that

$$\big\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\big\} \subseteq \big\{b_j + B_j\alpha \mid \alpha \in \mathbf{1}\big\}. \tag{8.7}$$

This can be done as follows.

Let $H_j \in \mathbb{R}^{n\times n}$ be nonsingular and denote

$$\boldsymbol{r}_j = (H_j^{-1}C_j)\,\mathbf{1} + H_j^{-1}\boldsymbol{e}_j \quad \text{and} \tag{8.8}$$

$$D_j = \mathrm{diag}\big(\mathrm{w}(\boldsymbol{r}_j)\big). \tag{8.9}$$

The choice of the transformation matrix H_j is discussed in the next subsection. We have

$$\begin{aligned}
&\left\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\right\} \\
&\qquad = \left\{c_j^{(0)} + H_j\big((H_j^{-1}C_j)\alpha + H_j^{-1}e\big) \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\right\} \\
&\qquad \subseteq \left\{c_j^{(0)} + H_j r \mid r \in \boldsymbol{r}_j\right\} \qquad (8.10)\\
&\qquad = \left\{c_j^{(0)} + H_j\underline{\boldsymbol{r}}_j + H_j r \mid r \in [0, \overline{\boldsymbol{r}}_j - \underline{\boldsymbol{r}}_j] = D_j\,\mathbf{1}\right\} \\
&\qquad = \Big\{\big(c_j^{(0)} + H_j\underline{\boldsymbol{r}}_j\big) + (H_jD_j)\alpha \mid \alpha \in \mathbf{1}\Big\}.
\end{aligned}$$

Thus, we can compute

$$b_j = c_j^{(0)} + H_j\underline{\boldsymbol{r}}_j \quad \text{and} \qquad (8.11)$$

$$B_j = H_jD_j. \qquad (8.12)$$

If we evaluate b_j and B_j by (8.11, 8.12) in floating-point arithmetic, due to roundoff errors, (8.7) may not hold. Subsection 8.3 presents an approach for ensuring that the computed b_j and B_j in floating-point arithmetic are such that (8.7) holds.

8.2. Choosing a transformation matrix. Similar to section 4, we wish to find a nonsingular matrix $H_j \in \mathbb{R}^{n\times n}$ such that

$$\left\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\right\} \subseteq \left\{c_j^{(0)} + H_j r \mid r \in \boldsymbol{r}_j\right\},$$

and $\left\{c_j^{(0)} + H_j r \mid r \in \boldsymbol{r}_j\right\}$ does not contain a large overestimation of $\left\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\right\}$. As discussed before, we can choose, for example,

- $H_j = C_j$ or
- $H_j = Q_j$, where $C_jP_j = Q_jR_j$.
  Here, P_j is a permutation matrix such that the columns of C_jP_j are sorted in non-increasing order of their lengths, Q_j is orthonormal, and R_j is upper triangular.

In this section, we also refer to the resulting methods as the P ($H_j = C_j$) and QR ($H_j = Q_j$) methods, respectively.

We have again parallelepiped- and QR-type enclosures, which carry the advantages and disadvantages discussed in section 4. However, here, we include $\boldsymbol{e}_j$ in the bounds on the solution set, while in the previous methods, $\boldsymbol{e}_j$ is included in $\boldsymbol{r}_j$ in (3.19), which is normally very small. Hence, in those methods, overestimations due to wrapping usually remain small, while in the present approach, they may not be very small; compare the overestimations in Figures 4.2–4.6 with the overestimations in Figure 8.2 from the next example.

We could keep two representations: one corresponding to the selection $H_j = C_j$ and the other to the choice $H_j = Q_j$. However, this would require performing $2(n+1)$ integrations with point initial conditions. Alternatively, we may switch between these two choices and keep one representation, and therefore, have $(n+1)$ point integrations.

This can be accomplished as described below. Denote

$$\boldsymbol{r}_{\mathrm{P},j} = \mathbf{1} + C_j^{-1}\boldsymbol{e}_j \quad \text{and} \qquad (8.13)$$

$$\boldsymbol{r}_{\mathrm{QR},j} = (Q_j^{-1}C_j)\,\mathbf{1} + Q_j^{-1}\boldsymbol{e}_j. \qquad (8.14)$$

Denote also the volume of $\{Q_j r \mid r \in r_{\mathrm{QR},j}\}$ by $V_{\mathrm{QR},j}$ and the volume of $\{C_j r \mid r \in r_{\mathrm{P},j}\}$ by $V_{\mathrm{P},j}$. A possible (heuristic) strategy for switching between the P and QR methods is to select

$$\begin{aligned} \text{QR method} \quad & \text{if } Q_j \boldsymbol{r}_{\mathrm{QR},j} \subseteq C_j \boldsymbol{r}_{\mathrm{P},j} \text{ or } V_{\mathrm{QR},j} \le V_{\mathrm{P},j} \\ \text{P method} \quad & \text{otherwise.} \end{aligned}$$

EXAMPLE 8.1. Consider

$$\begin{aligned} y' = Ay = \begin{pmatrix} 1 & -2 \\ 3 & -4 \end{pmatrix} y, \\ y(0) \in ([1,3],[1,3])^T. \end{aligned}$$

We apply four steps of the P and QR methods to this problem using constant stepsize $h = 0.3$. For the visualizations that follow, we set "artificially"

$$\boldsymbol{e}_j^{(i)} = ([-10^{-2}, 10^{-2}], [-10^{-2}, 10^{-2}])^T \quad \text{for all } j = 1,2,3,4 \text{ and all } i = 0,1,2.$$

That is, we assume that $\boldsymbol{e}_j^{(i)}$ contains the excess in each point solution on each step.

The enclosures resulting from these methods are shown in Figure 8.2. On steps 1 and 2, the P method should be applied, and on step 4, the QR method should be applied. Although $V_{\mathrm{P},3} < V_{\mathrm{QR},3}$ on step 3, the QR method is preferable, which can be clearly seen from Figure 8.3. In this case, $c_3^{(0)} + Q_3 \boldsymbol{r}_{\mathrm{QR},3} \subset c_3^{(0)} + C_3 \boldsymbol{r}_{\mathrm{P},3}$, and at the same time, $V_{\mathrm{P},3} < V_{\mathrm{QR},3}$.

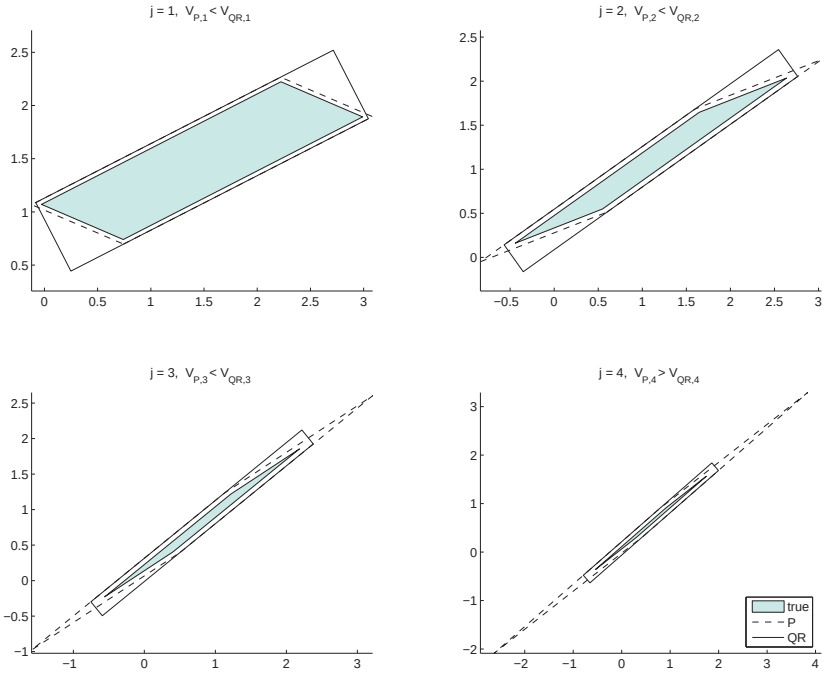


FIG. 8.2. *Enclosures computed by the P and QR methods. The true solution sets are the filled parallelepipeds.*

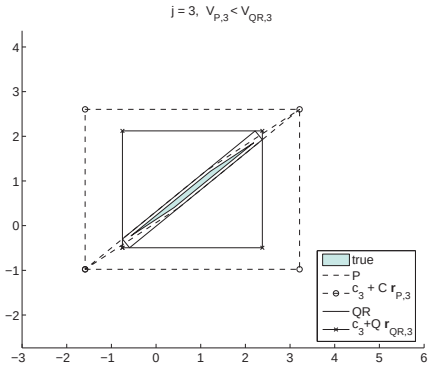


FIG. 8.3. *Enclosures computed by the P and QR methods.*

8.3. Including roundoff errors. It is not difficult to show

PROPOSITION 8.2. *Let $a, b \in \mathbb{R}^n$, $A, B \in \mathbb{R}^{n\times n}$, and $\boldsymbol{e} \in \mathbb{IR}^n$. If B is nonsingular, and if*

$$B^{-1}(a - b + \boldsymbol{e}) + (B^{-1}A)\,\mathbf{1} \subseteq \mathbf{1},$$

then

$$\{a + A\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}\} \subseteq \{b + B\alpha \mid \alpha \in \mathbf{1}\}.$$

In practice, we would compute floating-point approximations to b_j and B_j using (8.11, 8.12). Denoting these approximations by $\widetilde{b}_j$ and $\widetilde{B}_j$, respectively, we have to check if

$$\{c_j^{(0)} + C_j\alpha + e \mid \alpha \in \mathbf{1},\, e \in \boldsymbol{e}_j\} \subseteq \{\widetilde{b}_j + \widetilde{B}_j\alpha \mid \alpha \in \mathbf{1}\}; \tag{8.15}$$

cf. (8.7).

Using Proposition 8.2, we can evaluate in interval computer arithmetic

$$\boldsymbol{x}_j := \widetilde{B}_j^{-1}\big(c_j^{(0)} - \widetilde{b}_j + \boldsymbol{e}\big) + (\widetilde{B}_j^{-1}C_j)\,\mathbf{1}. \tag{8.16}$$

If $\boldsymbol{x}_j \subseteq \mathbf{1}$, then (8.15) holds. If $\boldsymbol{x}_j \not\subseteq \mathbf{1}$, then we can inflate $\boldsymbol{r}_j$, compute new values for b_j and B_j, new $\boldsymbol{x}_j$, and check if $\boldsymbol{x}_j \subseteq \mathbf{1}$. (For inflating techniques in interval computing, see [10, 25].)

This inflation can be carried out by adding $\boldsymbol{l}_j \in \mathbb{IR}^n$ with ith component $\boldsymbol{l}_{j(i)} \ni 0$ and $\mathrm{w}\big(\boldsymbol{l}_{j(i)}\big) > 0$, for all $i = 1, \ldots, n$, to $\boldsymbol{r}_j$:

$$\boldsymbol{z_j} = \boldsymbol{r}_j + \boldsymbol{l_j}, \tag{8.17}$$

where $\boldsymbol{r}_j$ is defined in (8.8).

Denote

$$\widehat{b}_j = c_j^{(0)} + H_j\underline{\boldsymbol{z}}_j, \tag{8.18}$$

$$\widehat{D}_j = \mathrm{diag}\,\big(\mathrm{w}(\boldsymbol{z}_j)\big), \quad \text{and} \tag{8.19}$$

$$\widehat{B}_j = H_j\widehat{D}_j. \tag{8.20}$$

Using (8.8, 8.17, 8.18, 8.20), it can be shown that

$$\begin{aligned}\widehat{\boldsymbol{x}} &:= \widehat{B}_j^{-1}(c_j^{(0)} - \widehat{b}_j + \boldsymbol{e}) + (\widehat{B}_j^{-1}C_j)\,\mathbf{1}\\ &= \widehat{D}_j^{-1}[-\underline{\boldsymbol{l}}_j, \mathrm{w}(\boldsymbol{r}_j) - \underline{\boldsymbol{l}}_j].\end{aligned}$$

Since by (8.17, 8.19), $\widehat{D}_j = \mathrm{diag}\left(\mathrm{w}(\boldsymbol{r}_j) + \mathrm{w}(\boldsymbol{l}_j)\right)$, the width of the ith component of $\widehat{\boldsymbol{x}}$ is

$$\mathrm{w}\big(\widehat{\boldsymbol{x}}_i\big) = \frac{\mathrm{w}\big(\boldsymbol{r}_{j(i)}\big)}{\mathrm{w}\big(\boldsymbol{z}_{j(i)}\big)} = \frac{\mathrm{w}\big(\boldsymbol{r}_{j(i)}\big)}{\mathrm{w}\big(\boldsymbol{r}_{j(i)}\big) + \mathrm{w}\big(\boldsymbol{l}_{j(i)}\big)} < 1.$$

Furthermore, since $-\underline{\boldsymbol{l}}_{j(i)} > 0$, $\widehat{\boldsymbol{x}}_i \subseteq [0,1]$. Therefore, we can make $\mathrm{w}(\widehat{\boldsymbol{x}}_i)$ as small as we wish by increasing $\mathrm{w}(\boldsymbol{l}_{j(i)})$. If the computed in floating-point interval arithmetic $\widehat{\boldsymbol{x}} \not\subseteq \mathbf{1}$, we can increase $\mathrm{w}(\boldsymbol{l})$, evaluate again $\widehat{b}_j$, $\widehat{B}_j$, and $\widehat{\boldsymbol{x}}$, and check if $\widehat{\boldsymbol{x}} \subseteq \mathbf{1}$.

REMARK 8.1. Choosing $\boldsymbol{l}_j$ such that $\mathrm{w}(\boldsymbol{l}_j)$ is as small as possible, and $\widehat{\boldsymbol{x}} \subseteq \mathbf{1}$ is satisfied on the first attempt, is a subtle task. We omit the details about how it can be accomplished.

Numerical experience (not reported here) suggest that this method generally works well. However, on some problems, the overestimations on the solution set may be larger than desired. This is due to applying anti-wrapping strategies, such as the P or QR approaches, on an enclosure of the solution set. If it is not sufficiently small, these overestimations may not be small either. For example, even if the $\boldsymbol{e}_j^{(i)}$ in Example 8.1 are of the order of the machine precision, the QR method could still introduce non-trivial overestimations; see Figure 8.2.

From an implementation perspective, ensuring that roundoff errors are included, as discussed in the previous subsection, makes the current method somewhat more cumbersome to implement than the QR-P method from section 4.

9. Concluding remarks. The methods we present for computing bounds on the solution of a linear IVP ODE have two important advantages: (1) they do not restrict the size of the initial box, and (2) they require only a method for enclosing point solutions. As a consequence, we avoid computing Taylor coefficients for the solution to the associated variational equation, which is normally done in an interval method for general ODEs. From a practical point of view, we do not require AD facilities for computing these coefficients.

The proposed QR-P approach reduces the wrapping effect more effectively than each of the P and QR methods. On nonlinear problems, it is expected that the QR-P method will produce tighter bounds that the QR method. Numerical studies in this case are part of future work.

To the author's experience, the QR-P approach is superior to the approach of following vertices. In the former, the excess is included in a local error term, which is normally very small; in the latter, the excess is included into the enclosures on the solution, which may not be very small. Moreover, the former approach is simpler to implement than the latter.

Finally, the presented techniques for constructing a parallelepiped containing $(n+1)$, n-dimensional boxes and enclosing roundoff errors in this parallelepiped may have applications in other problems, as for example in a convex hull computation.

Acknowledgments. The derivation (8.10) is by Rudolf Lohner, 2001, private communications. Ken Jackson made valuable comments on a late draft of this paper.

Qiang Song [26] implemented and studied empirically the method of following the vertices of a parallelepiped. Alexandre Korobkine (as a summer student in 2001) implemented and tested an algorithm based on the theory in subsections 8.1–8.3.

REFERENCES

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] C. Bendsten and O. Stauning, *FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods*, Tech. Rep. 1996-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, August 1996.

[3] ———, *TADIFF, a flexible C++ package for automatic differentiation using Taylor series*, Tech. Rep. 1997-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, April 1997.

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

[5] J. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley, 2003.

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

[7] P. Eijgenraam, *The Solution of Initial Value Problems Using Interval Arithmetic*, Mathematical Centre Tracts No. 144, Stichting Mathematisch Centrum, Amsterdam, 1981.

[8] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in applied mathematics, SIAM, Philadelphia, PA, 2000.

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

[10] V. Kreinovich, S. Starks, and G. Mayer, *On a theoretical justification of the choice of epsilon-inflation in PASCAL-XSC*, Reliable Computing, 3 (1997), pp. 437–445.

[11] M. Lerch, G. Tischler, and J. Wolff von Gudenberg, *filib++—interval library specification and reference manual*, Tech. Rep. 279, Universität Würzburg, Germany, 2001.

[12] R. J. Lohner, *Anfangswertaufgaben im kompakten Mengen für Anfangswerte und Parameter*, diplomarbeit, Inst. f. Angew. Math., Universität Karlsruhe, 1978.

[13] ———, *Einschließung der Lösung gewöhnlicher Anfangs– und Randwertaufgaben und Anwendungen*, PhD thesis, Universität Karlsruhe, 1988.

[14] K. Makino and M. Berz, *Suppression of the wrapping effect by Taylor model-based validated integrators*, Tech. Rep. MSU HEP 40910, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA, 2004.

[15] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1966.

[16] N. S. Nedialkov, *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*, PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, M5S 3G4, February 1999.

[17] N. S. Nedialkov and K. R. Jackson, *An interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation*, Reliable Computing, 5 (1999), pp. 289–310. Also in T. Csendes, editor, *Developments in Reliable Computing*, pp. 289–310, Kluwer, Dordrecht, Netherlands, 1999.

[18] N. S. Nedialkov and K. R. Jackson, *A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations*, in Perspectives on Enclosure Methods, A. Facius, U. Kulisch, and R. Lohner, eds., Springer-Verlag, Vienna, 2001, pp. 219–264.

[19] N. S. Nedialkov and K. R. Jackson, *The design and implementation of a validated object-oriented solver for IVPs for ODEs*, Tech. Rep. 6, Software Quality Research Laboratory, Department of Computing and Software, McMaster University, Hamilton, Canada, L8S 4L7, 2002.

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

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

[22] K. Nickel, *How to fight the wrapping effect*, in Interval Analysis 1985, K. Nickel, ed., Lecture Notes in Computer Science No. 212, Springer, Berlin, 1985, pp. 121–132.
[23] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, vol. 120 of Lecture Notes in Computer Science, Springer Verlag, Berlin, 1981.
[24] R. Rihm, *On a class of enclosure methods for initial value problems*, Computing, 53 (1994), pp. 369–377.
[25] S. M. Rump, *A note on epsilon-inflation*, Reliable Computing, 4 (1998), pp. 371–375.
[26] Q. Song, *Computing tight bounds on the solution of an initial value problem for a linear differential equation*, Master's thesis, Dept. of Computing and Software, McMaster University, Hamilton, Ontario, Canada, L8S 4L7, February 2004.
[27] O. Stauning, *Automatic Validation of Numerical Solutions*, PhD thesis, Technical University of Denmark, DK-2800, Lyngby, Denmark, October 1997.
[28] H. Stetter, *Tolerance proportionality in ODE–codes*, in Proc. Second Conf. on Numerical Treatment of Ordinary Differential Equations, R. M arz, ed., Humbold University, Berlin, 1980, pp. 109–123.
[29] H. J. Stetter, *Validated solution of initial value problems for ODEs*, in Computer Arithmetic and Self-Validating Numerical Methods, C. Ullrich, ed., Academic Press, New York, 1990, pp. 171–187.
[30] G. Walster, E. Hansen, and J. Pryce, *Extended real intervals and the topological closure of extended real relations*, tech. rep., Sun Microsystems, February 2000.