ELSEVIER

Applied Numerical Mathematics 20 (1996) 21-37

APPLIED NUMERICAL MATHEMATICS

Analysis and implementation of TR-BDF2

M.E. Hosea [a,*], L.F. Shampine [b,1]

[a] *Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL 60115-2888, USA*

[b] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275-0156, USA*

Abstract

Bank et al. (1985) developed a one-step method, TR-BDF2, for the simulation of circuits and semiconductor devices based on the trapezoidal rule and the backward differentiation formula of order 2 that provides some of the important advantages of BDF2 without the disadvantages of a memory. Its success and popularity in the context justify its study and further development for general-purpose codes. Here the method is shown to be strongly S-stable. It is shown to be optimal in a class of practical one-step methods. An efficient, globally C^1 interpolation scheme is developed. The truncation error estimate of Bank et al. (1985) is not effective when the problem is very stiff. Coming to an understanding of this leads to a way of correcting the estimate and to a more effective implementation. These developments improve greatly the effectiveness of the method for very stiff problems.

1. Introduction

Because one-step methods have advantages for the solution of systems of ordinary differential equations arising in circuit and device simulation, the trapezoidal rule is the default method in popular packages like SPICE [22]. As usually implemented, the trapezoidal rule is not strictly a one-step method because the truncation error estimate makes use of two previously computed solution values. The method is not efficient for very stiff problems because it is not strongly stable. Many simulation packages resort to methods with memory, specifically the backward differentiation formulas, to obtain the strong stability necessary for the efficient solution of such problems. In the context of device simulation, Bank et al. [2] developed a method that provides the advantages of the second order backward differentiation formula, BDF2, without the disadvantages of a memory. A conventional integration with BDF2 is started with one step taken with a one-step method of order at least one. They noted that if this is followed by a second step taken with BDF2, the whole process can be

* Corresponding author. E-mail: hosea@na-net.ornl.gov.

[1] E-mail: shampine@na-net.ornl.gov.

repeated cyclically to obtain a method with two internal steps and no memory. It is natural to use the trapezoidal rule for the first step because it provides a result of the same accuracy as BDF2 and has good stability. One of the things that makes a conventional BDF code efficient is the possibility of using the same simplified Newton iteration matrix for a number of steps. A key issue, then, is to arrange that this be true for the evaluation of the two different implicit formulas. This turns out to be possible when the two internal steps are of different lengths. The process might be described as a cyclic linear multistep method, LMM, but it is more naturally studied as a one-step method. Bank et al. showed their method, TR-BDF2, to be L-stable. They also found an error estimate that does not involve a memory, so that the method is genuinely one-step.

The success and popularity of TR-BDF2 in device simulation justify its study and further development for general-purpose codes. Moreover, BDF2 is widely used in contexts such as the method of lines (MOL) solution of partial differential equations (PDEs) where a variant that is one-step would be very useful. For this reason, Carroll [6] studies a variant of TR-BDF2 for the solution of parabolic PDEs in one space dimension by the MOL. By regarding TR-BDF2 and its error estimate as a Diagonally Implicit Runge-Kutta, DIRK, pair, we can bring to bear a considerable theory for the study of its stability. It proves to be illuminating to derive an analogous formula based on the trapezoidal rule. By taking two steps of the same size with the trapezoidal rule, we obtain another DIRK formula, TRX2, for which we can derive a one-step error estimate like that used for TR-BDF2. A variant of the trapezoidal rule that is more one-step than conventional implementations is of some independent interest.

As an implicit Runge-Kutta formula, TR-BDF2 is quite unusual because it involves an explicit stage. This complicates investigation of its stability because some of the standard theorems are not directly applicable. Among other results, we prove that TR-BDF2 is strongly S-stable. We also show that it is optimal in a class of computationally interesting one-step methods. In general-purpose codes, it is important that it be possible to achieve solution values between steps inexpensively. Indeed, we are developing a code that implements both TR-BDF2 and TRX2 for the MATLAB ODE suite [19]. Because event location is a feature of all the codes in this suite, any method implemented must provide for a continuous extension to the solution. We show how to obtain this for "free" with TR-BDF2 and TRX2 by means of globally C^1 piecewise polynomial interpolants.

The trapezoidal rule and BDF2 have truncation errors that are multiples of y'''. In conventional implementations this is estimated by a divided difference of approximate solution values, y_n. This requires approximations computed prior to the current step. To retain the one-step nature of TR-BDF2, the estimator of [2] forms a divided difference of approximations to the first derivative of the solution, $f_n = f(x_n, y_n)$. Although this has proved acceptable in practice, these approximations are not used in conventional BDF codes because it is well known [17] that they do not provide accurate approximations to the first derivative when the problem is very stiff. When we investigate TR-BDF2 and TRX2 in a manner usual for one-step methods, we view each step as being taken with two formulas and the error of the lower order formula estimated by comparison to the higher. The difficulty with the error estimate of [2] is then revealed in the poor stability of the companion formula of order 3. Once we understand the difficulties, we appreciate that if TR-BDF2 is implemented in the obvious way, the error estimator of [2] will limit severely the efficiency of the formula when integrating very stiff problems. After coming to an understanding of the difficulty, we develop practical remedies for both TR-BDF2 and TRX2.

2. TR-BDF2 as a DIRK method

The TR-BDF2 method and associated error estimate derived by Bank et al. [2] can be viewed as an implicit Runge–Kutta pair of orders 2 and 3, more specifically, a Diagonally Implicit Runge–Kutta, DIRK, 2(3) pair:

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\gamma & d & d & 0 \\
1 & w & w & d \\
\hline
 & w & w & d \\
\hline
 & (1-w)/3 & (3w+1)/3 & d/3
\end{array}$$

where $\gamma = 2 - \sqrt{2}$, $d = \gamma/2$, and $w = \sqrt{2}/4$. We use the standard terminology for Butcher arrays

$$\begin{array}{c|c}
c & A \\
\hline
 & b^{\mathrm{T}} \\
\hline
 & \widehat{b}^{\mathrm{T}}
\end{array}$$

Singly Diagonally Implicit Runge–Kutta (SDIRK) methods, i.e., DIRK methods for which the diagonal elements of A are the same, are attractive because all the implicit stages in a step can be evaluated using the same simplified Newton iteration matrix. TR-BDF2 is *almost* an SDIRK method. The diagonal elements of A are the same except for the first. However, since the first stage is explicit, there is no nonlinear equation to be solved for its evaluation. Moreover, the formula is First-Same-As-Last, FSAL, meaning that the first stage of a step is the same as the last stage from the end of the previous step.

In a certain sense, TR-BDF2 is optimal in the class of 3-stage DIRK methods. To fix notation, let the general 3-stage DIRK method be given by

$$\begin{array}{c|ccc}
c_1 & a_{1,1} & 0 & 0 \\
c_2 & a_{2,1} & a_{2,2} & 0 \\
c_3 & a_{3,1} & a_{3,2} & a_{3,3} \\
\hline
 & b_1 & b_2 & b_3 \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3
\end{array}$$

As we see it, the advantages of TR-BDF2 are:

(i) It is First-Same-As-Last (FSAL), hence there are only two implicit stages to evaluate per step, not three.

(ii) The same simplified Newton iteration matrix can be used to evaluate all the implicit stages.

(iii) It provides a "free" asymptotically correct error estimate.

(iv) It is L-stable (in fact, strongly S-stable).

(v) All the stages are evaluated within the step interval, i.e., $0 \leqslant c_i \leqslant 1$, for $i = 1, 2, 3$.

We do not want to give up any of these properties. The first two require the general 3-stage DIRK method to have the form

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
c_2 & a_{2,1} & b_3 & 0 \\
1 & b_1 & b_2 & b_3 \\
\hline
 & b_1 & b_2 & b_3 \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3
\end{array}$$

For the third, the primary result must satisfy the order conditions up to order two and the companion formula, the order conditions up to order three. In the formulation of [11], this is

$$1 - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{e} = 0 \Longrightarrow b_1 + b_2 + b_3 = 1,$$

$$\tfrac{1}{2} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{c} = 0 \Longrightarrow b_2c_2 + b_3 = \tfrac{1}{2},$$

$$1 - \widehat{\boldsymbol{b}}^{\mathrm{T}}\boldsymbol{e} = 0 \Longrightarrow \widehat{b}_1 + \widehat{b}_2 + \widehat{b}_3 = 1,$$

$$\tfrac{1}{2} - \widehat{\boldsymbol{b}}^{\mathrm{T}}\boldsymbol{c} = 0 \Longrightarrow \widehat{b}_2c_2 + \widehat{b}_3 = \tfrac{1}{2},$$

$$\tfrac{1}{6} - \tfrac{1}{2}\widehat{\boldsymbol{b}}^{\mathrm{T}}\boldsymbol{c}^2 = 0 \Longrightarrow \widehat{b}_2c_2^2 + \widehat{b}_3 = \tfrac{1}{3},$$

$$\widehat{\boldsymbol{b}}^{\mathrm{T}}\left(\tfrac{1}{2}\boldsymbol{c}^2 - A\boldsymbol{c}\right) = 0 \Longrightarrow \widehat{b}_2c_2\left(\tfrac{1}{2}c_2 - b_3\right) + \widehat{b}_3\left(\tfrac{1}{2} - b_2c_2 - b_3\right) = 0.$$

Consistency also requires that $c_2 = a_{2,1} + b_3$. It is easy to show that this system of equations leads to the one-parameter family

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\theta & \dfrac{\theta}{2} & \dfrac{\theta}{2} & 0 \\
1 & \dfrac{3\theta-\theta^2-1}{2\theta} & \dfrac{1-\theta}{2\theta} & \dfrac{\theta}{2} \\
\hline
 & \dfrac{3\theta-\theta^2-1}{2\theta} & \dfrac{1-\theta}{2\theta} & \dfrac{\theta}{2} \\
\hline
 & \dfrac{3\theta-1}{6\theta} & \dfrac{1}{6\theta(1-\theta)} & \dfrac{2-3\theta}{6(1-\theta)}
\end{array}$$

The stability function for the primary result in this family is

$$\begin{aligned}
R_2(h\lambda) &= 1 + h\lambda\boldsymbol{b}^{\mathrm{T}}(I - h\lambda A)^{-1}\boldsymbol{e} \\
&= \frac{(\theta^2 - 4\theta + 2)(h\lambda)^2 + 4(1-\theta)h\lambda + 4}{(2 - \theta h\lambda)^2} \\
&\longrightarrow \frac{(\theta^2 - 4\theta + 2)}{\theta^2} \quad \text{as } |h\lambda| \to \infty.
\end{aligned}$$

Here I and $\boldsymbol{e}$ denote the identity matrix and vector of all ones, respectively, of appropriate size. The method is L-stable only if $\theta^2 - 4\theta + 2 = 0$. Thus, $\theta = 2 \pm \sqrt{2}$. Since $2 + \sqrt{2} > 1$, the final advantage requires that $\theta = 2 - \sqrt{2}$, which leads to TR-BDF2.

It is clear from this discussion that no second order DIRK method of fewer stages can have all the desired properties. In particular, the second order formula from [1]

$$\begin{array}{c|cc} d & d & 0 \\ 1 & 1-d & d \\ \hline & 1-d & d \end{array}$$

is strongly S-stable, but admits no embedded third order companion, even with the addition of a FSAL stage. In his numerical tests, Alexander estimates the local error by doubling. This provides a quality estimate, but is expensive in several respects and makes the formula much less attractive in practice than its good stability properties would suggest.

In [5,6], the trapezoidal rule in TR-BDF2 is replaced by the θ method

$$y_{n+\gamma} = y_n + \gamma h[(1-\theta)f(x_n, y_n) + \theta f(x_{n+\gamma}, y_{n+\gamma})], \quad 0 < \theta \leqslant 1.$$

The resulting family of Runge–Kutta formulas forms a subset of those considered here. Unfortunately, the only member of this family that admits an embedded, asymptotically correct error estimate is TR-BDF2 itself. In particular, the error estimate of [5] is not asymptotically correct unless $\theta = 1/2$.

The choice $\theta = 1/2$ in our one-parameter family results in the pair

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ 1 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \hline & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Because the primary result here is equivalent to a double step of the trapezoidal rule, we call this method TRX2. It shares most of the advantages of TR-BDF2, but lacks L-stability (it is not damped at infinity).

When L-stability is not needed, we expect TRX2 to be somewhat more efficient than TR-BDF2 in terms of evaluations of f because it has a smaller truncation error. Using the approach of [11], it is found that (asymptotically) TRX2 is roughly 25% more efficient than TR-BDF2. We do see this in practice. Of course, when stiffness is severe, L-stability is crucial and TR-BDF2 enjoys an enormous advantage over TRX2.

3. Stability

It is shown in [2] that TR-BDF2 is L-stable. A number of other stability concepts have been identified as being useful for the study of implicit Runge–Kutta methods, and we ask which, if any, TR-BDF2 possesses. The first theorem proves negative results for a class of formulas that includes TR-BDF2, and the second proves that the method is strongly S-stable.

Theorem 1. *Consider an FSAL DIRK method of s stages*

$$\begin{array}{c|c} c & A \\ \hline & b^{\mathrm{T}} \end{array}$$

where all the elements of $\boldsymbol{c}$ *are distinct (i.e. the method is nonconfluent,* [9]) *and the stages have been arranged so that the first row of* A *is zero. If* $b_1 \neq 0$, *then the method is neither AN-stable nor B-stable nor algebraically stable.*

Proof. In the notation of [9, p. 196], let

$$K(Z) = 1 + \boldsymbol{b}^{\mathrm{T}} Z (I - AZ)^{-1} \boldsymbol{e},$$

where $Z = \operatorname{diag}(z_1, \dots, z_s)$ with $\operatorname{Re}(z_i) \leqslant 0$. Let $z_i = 0$, for $i \geqslant 2$. It follows directly that

$$K(Z) = 1 + b_1 z_1,$$

which is not bounded, hence the method is not AN-stable. Lack of B-stability and algebraic stability follow from [9, Corollary 12.14], which says that for nonconfluent Runge–Kutta methods, the concepts of AN-stability, B-stability, and algebraic stability are equivalent. □

Theorem 2. *TR-BDF2 is strongly S-stable.*

Proof. The A-stability of TR-BDF2 is shown in [2, p. 440]. In the notation of [14], let

$$\alpha_0 = \lim_{|z| \to 0} \left(1 - \boldsymbol{b}^{\mathrm{T}} (A - zI)^{-1} \boldsymbol{e}\right) \quad \text{and} \quad \boldsymbol{b}_0^{*\mathrm{T}} = \lim_{|z| \to 0} \boldsymbol{b}^{\mathrm{T}} (A - zI)^{-1} E(z),$$

where in our case $E(z) = \operatorname{diag}(-z, \gamma, 1)$ and the limits are taken with $\operatorname{Re}(z) < 0$. By direct evaluation we get

$$\alpha_0 = \lim_{|z| \to 0} \left(1 - \frac{d^2 - z}{(d - z)^2}\right) = 0$$

and

$$\boldsymbol{b}_0^{*\mathrm{T}} = \lim_{|z| \to 0} \left(\frac{wz^2}{(d - z)^2}, -\frac{w\gamma z}{(d - z)^2}, \frac{d}{d - z}\right)^{\mathrm{T}} = (0, 0, 1)^{\mathrm{T}}.$$

By [14, Theorem 2.1], it follows that TR-BDF2 is S-stable. Since we have $\alpha_0 = 0$, [14, Theorem 2.2] implies that the method is strongly S-stable if it is stiffly accurate, i.e. if

$$\lim_{|z| \to 0} \beta(z, G_0, \boldsymbol{G}_1, \boldsymbol{G}_2) = 0,$$

where in our case

$$G_0 = \frac{g(x_n + h) - g(x_n)}{h},$$

$$\boldsymbol{G}_1 = \left(g'(x_n), \frac{g(x_n + \gamma h) - g(x_n)}{\gamma h}, G_0\right)^{\mathrm{T}},$$

$$G_2 = \left(0, g'(x_n+\gamma h), g'(x_n+h)\right)^{\mathrm{T}},$$

and

$$\beta(z, G_0, \boldsymbol{G}_1, \boldsymbol{G}_2) = -G_0 + \boldsymbol{b}^{\mathrm{T}}(A - zI)^{-1}E(z)(\boldsymbol{G}_1 - z\boldsymbol{G}_2).$$

Here we are using the scalar test problem $y' = g'(x) + \lambda(y - g(x))$. After some tedious calculation, it is found that

$$\beta(z, G_0, \boldsymbol{G}_1, \boldsymbol{G}_2) = \frac{z^2 v_1(x_n) + z v_2(x_n)}{(d-z)^2},$$

where

$$v_1(x_n) = wg'(x_n) + w\gamma g'(x_n+\gamma h) + dg'(x_n+h) - G_0,$$

and

$$v_2(x_n) = dG_0 - w\gamma\frac{g(x_n+\gamma h) - g(x_n)}{h\gamma} - d^2 g'(x_n+h).$$

Since $v_1(x_n)$ and $v_2(x_n)$ are independent of z, $\lim_{|z|\to 0}\beta(z, G_0, \boldsymbol{G}_1, \boldsymbol{G}_2) = 0$, and the result follows. □

Because TRX2 can be viewed as a double step of the trapezoidal rule, which is otherwise known as the second order Lobatto IIIA formula, [14, Theorem 4.4] implies that TRX2 is S(α)-stable with $\alpha \in (0, \pi/2)$.

4. Interpolation

Extending an implicit Runge–Kutta method so as to obtain accurate solution values between steps is generally not obvious and often not possible. However, the origin of TR-BDF2 suggests an easy way to obtain these values. At each step TR-BDF2 provides three solution values and three derivative values that are all second order accurate. Let $x_{n+\gamma} = x_n + \gamma h$ and $x_{n+1} = x_n + h$. Correspondingly, let y_n, $y_{n+\gamma}$, and y_{n+1} denote the approximations to $y(x_n)$, $y(x_{n+\gamma})$, and $y(x_{n+1})$, respectively, and let z_n, $z_{n+\gamma}$, and z_{n+1} denote the approximations to $hy'(x_n)$, $hy'(x_{n+\gamma})$, and $hy'(x_{n+1})$, respectively. It is natural to obtain intermediate values by interpolating these approximations. Indeed, there are numerous possibilities. A particularly attractive option is to use cubic Hermite interpolation, as it is efficient and results in a globally C^1 interpolation scheme. The interpolant is given by

$$P(x) = (v_3 - 2v_2)r^3(x) + (3v_2 - v_3)r^2(x) + v_1 r(x) + v_0,$$

where, for $x_n \leqslant x \leqslant x_{n+\gamma}$,

$$v_0 = y_n, \quad v_1 = \gamma z_n, \quad v_2 = y_{n+\gamma} - y_n - v_1,$$

$$v_3 = \gamma(z_{n+\gamma} - z_n), \quad \text{and} \quad r(x) = \frac{x - x_n}{\gamma h},$$

and, for $x_{n+\gamma} \leqslant x \leqslant x_{n+1}$,

$$v_0 = y_{n+\gamma}, \quad v_1 = (1-\gamma) z_{n+\gamma}, \quad v_2 = y_{n+1} - y_{n+\gamma} - v_1,$$

$$v_3 = (1-\gamma)(z_{n+1} - z_{n+\gamma}), \quad \text{and} \quad r(x) = \frac{x - x_{n+\gamma}}{(1-\gamma)h}.$$

5. Evaluation

Let $h = h_n$ denote the current step size. The definition of the formula includes the explicit stage $z_n = hf(x_n, y_n)$. At the start or at a restart, the stage must be formed in this way, and it can be formed in this way at any step. A less obvious way to form the stage after a step has been taken is to obtain it from a simple rescaling of the value of z computed as the last stage in the previous step,

$$z_n = (h/h_{n-1}) z_{(n-1)+1}.$$

In common terminology, the obvious way of proceeding evaluates the last stage of a step in $P(EC)^k$ mode, and an additional evaluation E is made as it is used at the beginning of the next step. (The number of corrections k is not fixed.) Thus, the stage at the beginning of the step is computed by a $P(EC)^kE$ scheme. Evaluated in this way, the formula is not FSAL. As Nørsett and Thomsen [13] point out, $P(EC)^kE$ schemes are preferred for nonstiff problems and for stiff problems $P(EC)^k$ schemes are better. This is easily understood [17, pp. 416–417]—the additional evaluation amplifies stiff components unless the stage has been evaluated very accurately. We look closely at this in Section 6.

For the initial guess of the first implicit stage, we take $z^0_{n+\gamma} = z_n$. This is a crude approximation, and ordinarily it is possible to obtain an initial guess by extrapolating the derivative of the interpolant from the end of the previous step. However, this would add overhead and introduce a form of memory. It might be worth the trouble if the average number of iterations were reduced substantially, but our experiments have shown that at loose tolerances it often results in a worse guess, leading to more iterations and more work. It is consistently helpful at more stringent tolerances, but such circumstances are beyond the recommended purview of second order methods.

For any k we take

$$y^k_{n+\gamma} = (y_n + dz_n) + dz^k_{n+\gamma}.$$

Then $z_{n+\gamma}$ is computed by the simplified Newton iteration

$$(I - hdJ)\Delta^k = hf(x_{n+\gamma}, y^k_{n+\gamma}) - z^k_{n+\gamma} \quad \text{and} \quad z^{k+1}_{n+\gamma} = z^k_{n+\gamma} + \Delta^k.$$

Here $J \approx \partial f/\partial y$. It is crucial that $z_{n+\gamma}$ be defined as the result of this iteration and not as $hf(x_{n+\gamma}, y_{n+\gamma})$ for reasons amplified in Section 6. The smoothed derivative $z_{n+\gamma}$ is computed to an accuracy of τ. How the accuracy is assessed is discussed in [16]. Below we discuss the selection of a suitable value for τ.

The initial guess for the final stage is obtained by extrapolating the derivative of the cubic Hermite interpolant to values at x_n and $x_{n+\gamma}$. This leads to

$$z^0_{n+1} = (1.5 + \sqrt{2}) z_n + (2.5 + 2\sqrt{2}) z_{n+\gamma} - (6 + 4.5\sqrt{2})(y_{n+\gamma} - y_n).$$

(The corresponding guess for TRX2 is $5z_n + 8z_{n+\gamma} - 24(y_{n+\gamma} - y_n)$.)

For any k we take

$$y_{n+1}^k = (y_n + wz_n + wz_{n+\gamma}) + dz_{n+1}^k,$$

and the simplified Newton iteration becomes

$$(I - hdJ)\Delta^k = hf(x_{n+1}, y_{n+1}^k) - z_{n+1}^k \quad \text{and} \quad z_{n+1}^{k+1} = z_{n+1}^k + \Delta^k.$$

As with the computation of $z_{n+\gamma}$, this quantity is computed to an accuracy of τ.

An important practical matter is deciding when to terminate the iteration for $z_{n+\gamma}$ and z_{n+1}. Conventionally this has been done when it appears that the quantities have been computed to a fraction κ of the tolerance ϵ placed on the local error. A value of $\kappa = 0.1$ is representative. Nørsett and Thomsen [13] build on their earlier work with Houbak [12] to deduce a reasonable value for κ from the coefficients of formula and its companion. Unfortunately, their work is not immediately applicable to our situation because of the explicit stage and our use of a smoothed derivative from a previous step. A reasonable way to select κ can be deduced easily. In terms of the scaled derivatives, the estimated local error is

$$\mathbf{est} = (\hat{b}_1 - b_1)z_n + (\hat{b}_2 - b_2)z_{n+\gamma} + (\hat{b}_3 - b_3)z_{n+1}.$$

If each of the smoothed derivatives is computed to an accuracy of $\tau = \kappa\epsilon$, this estimate will be evaluated with an error no greater than $(|\hat{b}_1 - b_1| + |\hat{b}_2 - b_2| + |\hat{b}_3 - b_3|)\kappa\epsilon$. Since the sum here is equal to $2/3$, if we take $\kappa = 1/2$, we can expect the estimate to be evaluated with an error no bigger than $\epsilon/3$, which should be small enough to judge adequately the success of the step.

6. Error estimation

The local error is estimated by comparing the result of the second order formula to that of the embedded third order formula. This provides an asymptotically correct estimate as the step size tends to zero, but it is not clear that the estimate is useful when the step size is not "small". The situation can be studied in a way that resembles closely the analysis of absolute stability. With the same care necessary when interpreting the results of an analysis of absolute stability, insight and useful guides for practice can be obtained. Although our attention is focussed on TR-BDF2 and a one-step estimate of its error, investigation of a corresponding estimate for TRX2 is illuminating and of some independent interest.

The stability of the companion used for error estimation is much less satisfactory than that of TR-BDF2. The same is true of TRX2. Specifically, the stability function for the second order result of TR-BDF2 applied to the scalar test equation $y' = \lambda y$ is

$$R_2(h\lambda) = \frac{1 + h\lambda(1-\gamma)}{(1 - dh\lambda)^2} \sim \frac{2(1+\sqrt{2})}{h\lambda} \quad \text{as } |h\lambda| \to \infty,$$

and the stability function of the companion is

$$R_3(h\lambda) = \frac{1 + 2(1-\gamma)h\lambda + \frac{2}{3}d^2(d-1)(h\lambda)^3}{(1 - dh\lambda)^2} \sim -\frac{\sqrt{2}}{3}h\lambda \quad \text{as } |h\lambda| \to \infty.$$

For TRX2, the stability function for the second order result is

$$\widehat{R}_2(h\lambda) = \left(\frac{4+h\lambda}{4-h\lambda}\right)^2 \sim 1 \quad \text{as } |h\lambda| \to \infty,$$

and the stability function for the companion is

$$\widehat{R}_3(h\lambda) = \frac{-(h\lambda)^3 + 3(h\lambda)^2 + 24(h\lambda) + 48}{3(h\lambda)^2 - 24(h\lambda) + 48} \sim -\frac{1}{3}h\lambda \quad \text{as } |h\lambda| \to \infty.$$

Other investigations have shown that a mismatch in behavior at infinity leads to an unsatisfactory error estimate when the problem is stiff and considered what might be done about it. The present situation has much in common with Scraton's [15] one-step estimate of the error of Wolfbrandt's [21] one-step W-method of order two. Chua and Dew [7] and Zedan [23] suggest a modification to Scraton's estimator to correct the behavior at infinity and so allow a code based on Wolfbrandt's formula to use a step size more appropriate to the formula. We present here a similar modification to the error estimator of [2] for TR-BDF2. Among implicit Runge–Kutta formulas, TR-BDF2 is very unusual because it has an explicit stage. We shall see that because of this, how the formula is evaluated can have a profound effect on the estimation of error and the cost of the integration.

As in the analysis of absolute stability, we restrict our attention to problems of the form

$$y' = Jy + g,$$

where g is a constant vector. The Jacobian J is a constant matrix that is assumed to have a complete set of eigenvectors with eigenvector v_i corresponding to eigenvalue λ_i. It is supposed that the eigenvalues are nonzero and that when accuracy permits a step size h, either $|h\lambda_i|$ is "small" or $\mathrm{Re}(\lambda_i) < 0$. This class of problems is simple enough that we can work out all the details needed to understand the behavior of a numerical method. We require that the method performs adequately on this class of equations and hope that its behavior will be similar for more general problems.

The assumptions imply the existence of a constant solution of the differential equation, namely $q = -J^{-1}g$. The local solution $u(x)$ is the solution with $u(x_n) = y_n$. Its value at $x_{n+1} = x_n + h$ is

$$u(x_n + h) = q + \exp(hJ)(y_n - q).$$

Because the methods being investigated are linear for linear problems and exact for the constant solution, the numerical solution obtained with TR-BDF2 satisfies

$$y_{n+1} = q + R_2(hJ)(y_n - q).$$

Similarly, the companion formula results in

$$y^*_{n+1} = q + R_3(hJ)(y_n - q).$$

The local error of the step is then

$$u(x_n + h) - y_{n+1} = (\exp(hJ) - R_2(hJ))\,(y_n - q),$$

and it is estimated by

$$\mathbf{est} = y^*_{n+1} - y_{n+1} = (R_3(hJ) - R_2(hJ))\,(y_n - q).$$

These expressions can be understood more easily when the vectors are written in terms of the basis of eigenvectors. To this end, suppose that

$$y_n - q = \sum_j \sigma_{j,n} v_j.$$

The local error then becomes

$$u(x_{n+1}) - y_{n+1} = \sum_j \left(\exp(h\lambda_j) - R_2(h\lambda_j)\right) \sigma_{j,n} v_j,$$

and the estimate becomes

$$\mathbf{est} = y^*_{n+1} - y_{n+1} = \sum_j \left(R_3(h\lambda_j) - R_2(h\lambda_j)\right) \sigma_{j,n} v_j.$$

As $h \to 0$,

$$\frac{R_3(h\lambda_j) - R_2(h\lambda_j)}{\exp(h\lambda_j) - R_2(h\lambda_j)} \to 1,$$

reflecting the fact that the estimator is asymptotically correct. The difficulty with the estimator of [2] is exposed when we ask about the "stiff" components, those for which $|h\lambda_j| \gg 1$. The factor $\exp(h\lambda_j) - R_2(h\lambda_j)$ in the local error behaves like $2(1+\sqrt{2})(h\lambda_j)^{-1}$, but the corresponding factor in the estimate behaves like $-(\sqrt{2}/3)(h\lambda_j)$. Clearly the error in the stiff components is grossly overestimated. The situation is similar with TRX2: The factor in the local error behaves like -1 and the corresponding factor in the estimate behaves like $-(h\lambda_j)/3$.

It is easy to modify the estimate to improve it for stiff components while preserving its accuracy as $h \to 0$. This is done by noting that the practical evaluation of the formula is accomplished by a simplified Newton iteration that involves repeated solution of linear equations using a factorization of a matrix $I - hdJ$. Let us define a modified error estimate **Est** as the solution of

$$(I - hdJ)\,\mathbf{Est} = \mathbf{est}.$$

This estimate is obtained at a modest cost because the matrix is already factored. The cost is especially low in the MATLAB computing environment because linear algebra is comparatively fast. In terms of the eigenvectors,

$$\mathbf{Est} = \sum_j \left(\frac{R_3(h\lambda_j) - R_2(h\lambda_j)}{1 - dh\lambda_j}\right) \sigma_{j,n} v_j.$$

It is clear from the definition that **Est** is asymptotically correct as $h \to 0$ because **est** is. Now, however, when $|h\lambda_j| \gg 1$, the factor for component j of the error estimate behaves like $\sqrt{2}/(3d)$, a much better approximation to the correct factor, though still not of the correct order. Because the discrepancy for TRX2 is not so strong, the process yields an error estimate with the correct order for this formula.

As pointed out in [18], the estimate for TR-BDF2 could be improved with the solution of another linear system, namely $(I - hdJ)\mathbf{EST} = \mathbf{Est}$, in order to get the correct behavior at infinity. Since this does not seem to be necessary in practice, we try now to understand why. It is the stiff components that are approximated very poorly. Whether this affects seriously the estimate of the local error depends on their relative size. For the model problem, the fact that TR-BDF2 is stiffly stable implies that these components are relatively small. This is not the case when integrating with TRX2 because it is not stiffly stable. To study the phenomenon for TR-BDF2, suppose that the step to x_n was of size h_{n-1}. From

$$y_n = q + R_2(h_{n-1}J)(y_{n-1} - q),$$

we see that

$$y_n - q = \sum_j \sigma_{j,n} v_j = \sum_j R_2(h_{n-1}\lambda_j)\sigma_{j,n-1} v_j,$$

hence that $\sigma_{j,n} = R_2(h_{n-1}\lambda_j)\sigma_{j,n-1}$. Because $R_2(h_{n-1}\lambda_j)$ behaves like $-2(1+\sqrt{2})(h_{n-1}\lambda_j)^{-1}$ and $|h_{n-1}\lambda_j| \gg 1$, the stiff components are heavily damped at each step.

If we were actually to solve a model problem with TR-BDF2, the stiff components would decay so rapidly that there would be little difference in the two error estimators. The question, then, is to what extent the model tells us what will happen for more general problems. It will turn out that how the formula is evaluated is crucial to this, but for the moment let us suppose that it is evaluated exactly. The argument then suggests that the effects of the stiff components will be damped strongly on all steps after the start, or restart, so that **est** is likely to be satisfactory, though perhaps not as accurate as **Est**. In contrast, when integrating with TRX2, the effects of stiff components are not damped out and modification of **est** is quite significant.

In the analysis, it is assumed that y_n satisfies exactly the formula for computing an approximate solution at x_n, and then the formula for the next step begins with the computation of $f(x_n, y_n)$. It is computationally convenient to work with the scaled derivative $z_n = hf(x_n, y_n)$ and henceforth we do so. This is what we want, but what we actually compute in the step from x_{n-1} to $x_n = x_{n-1} + h_{n-1}$ is a value $\hat{y}_n$and then we form $hf(x_n, \hat{y}_n)$. The scheme for evaluating the formula is to produce a value $\hat{y}_n$ that agrees well with y_n. However, this does not imply that the scaled derivative formed using it is close to the value we want. A mean value theorem gives

$$hf(x_n, \hat{y}_n) - hf(x_n, y_n) = h\mathcal{J}\,(\hat{y}_n - y_n),$$

where $\mathcal{J}$ is the Jacobian of f with entries evaluated at different arguments near (x_n, y_n). This shows that if the problem is stiff, the factor $h\mathcal{J}$ amplifies the stiff components. Since this excitation of the stiff components at each step counteracts the damping of these components provided by the stiff stability of the formula, modification of the error estimate is necessary to overcome it. When $\|h\mathcal{J}\|$ is of modest size, the problem is not stiff and either estimate might be used effectively. When we evaluate the formula exactly, as for example when the Jacobian is constant, the stiff components are not excited to a degree that necessitates modification of the estimate.

This matter is closely related to the issues discussed in [17, Section 2.3, Chapter 8]. The value y_n we want is the solution of an algebraic equation of the form $y_n = h_{n-1}\mu f(x_n, y_n) + \psi$. The reference cited discusses the variable to be used in the computation, but the key point is to define a scaled derivative by

$$\hat{z}_n = (\hat{y}_n - \psi)/\mu \approx h_{n-1} f(x_n, y_n).$$

After adjustment of the step size appearing here, this provides a convenient "smoothed" approximation to the desired scaled derivative. Now

$$(h/h_{n-1})\,\widehat{z}_n - hf(x_n, y_n) = (h/h_{n-1})\,(\widehat{y}_n - y_n)\,/\mu.$$

Since all quality solvers restrict the rate of increase of the step size for practical reasons, the difference between $\widehat{z}_n$ and the desired scaled derivative is at most a small multiple of the difference between the approximate solution and the desired solution. In particular, stiff components are not excited by the explicit stage. Of course, at the start and at restarts, we must form directly $hf(x_n, \widehat{y}_n)$, but it is clear that the formula is evaluated very much more accurately in this manner when the problem is stiff.

Because stiff components are not amplified when the smoothed scaled derivative is used, the damping provided by the stiffly stable formula TR-BDF2 allows the efficient solution of very stiff problems using **est**. It is valuable to use the smoothed scaled derivative with TRX2 as well, but that is not enough with this formula to permit the efficient solution of very stiff problems using **est**.

7. Numerical examples

Our experimental implementation of TR-BDF2 has a mixed error test with scalar relative and absolute error tolerances and a maximum norm. The experiments reported here all used a relative tolerance of `re` = .005 and an absolute tolerance of `ae` = 10^{-10}. The code requires an analytical Jacobian for the equation. It selects a starting step size automatically and attempts to choose the largest step size that will satisfy an error per step criterion with the specified tolerances. The user specifies which of the local error estimates is to be used in the integration. In either case, at each step both `e1` = ||**est**|| and `e2` = ||**Est**|| are computed, along with the local error, `le`. The local error of the step from x_n to $x_{n+1} = x_n + h$ is estimated by comparing y_{n+1} to a more accurate result computed by another integration with the tolerances 0.1 × `re`, 0.1 × `ae` and an initial step size of $h/3$. To assess the quality of the estimates, we computed both the maximum over all steps of `abs(e1-le)` and the mean, and similarly for `abs(e2-le)`. These quantities were then scaled by `re`. The measures of cost recorded are the number of successful steps, the number of step failures due to the estimated size of the local error, the number of step failures due to failure of the simplified Newton iteration to converge, the number of function (f) evaluations, the number of evaluations of the Jacobian $J = \partial f/\partial y$, the number of LU decompositions, and the number of linear system solutions. A new Jacobian is formed only when the Newton iteration fails and the current Jacobian is out of date. To assess the quality of the solution, the maximum and average local error, scaled by `re`, are reported.

All the sample calculations that follow were performed on a Sun SPARC10. We have done experiments integrating the equation

$$y' = \begin{pmatrix} -500 & 0 \\ 0 & -1 \end{pmatrix} y + \begin{pmatrix} 500\cos x - \sin x \\ \sin x + \cos x \end{pmatrix}$$

on the interval $[0, 12]$. The initial values $(1, 0)^{\mathrm{T}}$ yield the solution $(\cos x, \sin x)^{\mathrm{T}}$, and with this choice there is no initial transient. This example differs from the model problems only in that the inhomogeneous term is not constant. Since the implicit stages are evaluated exactly, the way the first stage of each step is evaluated is not at issue. Table 1 gives the results for this problem using **est** and **Est** with TR-BDF2 and TRX2. For both formulas the runs using **Est** were more efficient in every

Table 1
Sample problem # 1

	TR-BDF2		TRX2	
	est	**Est**	**est**	**Est**
Successful steps	52	40	44	33
Error failures	19	7	14	3
Iteration failures	0	0	0	0
Function evaluations	204	139	163	105
Jacobian evaluations	1	1	1	1
LU decompositions	60	43	45	31
Linear system solves	202	184	161	139
Maximum estimate error	0.66	0.12	0.94	0.22
Average estimate error	0.13	0.05	0.21	0.09
Maximum local error	0.61	0.75	0.54	0.79
Average local error	0.18	0.29	0.13	0.24

measure of cost, including the total number of linear systems solved—the extra expense of forming **Est** was more than justified by the savings derived through reducing the number of unnecessary step failures.

New phenomena appear if we allow J to vary. Now the way the first stage of each step is evaluated is important, and we have performed runs using a fully explicit first stage in each step ($z_n = hf(x_n, y_n)$), a procedure we call EFS, as well as using the smoothed first stage obtained by taking advantage of the FSAL property ($z_n = (h/h_{n-1})z_{(n-1)+1}$), which we refer to as SFS.

We consider the nonlinear test problem D4 of [8]

$$\begin{aligned} y_1' &= -0.013y_1 - 1000y_1y_3, \\ y_2' &= -2500y_2y_3, \\ y_3' &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3. \end{aligned}$$

It is to be solved on $[0, 50]$ with initial values $(1, 1, 0)^{\mathrm{T}}$. The results are given in Table 2. Integrations using **est** with EFS were very inefficient. The stiff components were excited by the evaluation of $f(x_n, y_n)$ at each step, and the resulting errors were reflected in exaggerated local error estimates. Since these typically cause gratuitous step failures, the resulting inefficient integrations are characterized by maximum estimate errors close to unity. It is revealing to inspect the behavior of **est** when the integration is controlled with **Est**. Then the maximum error in **est** observed using TR-BDF2 was 763.56, and the average was 125.70. The corresponding statistics for TRX2 were 771.20 and 132.63. However, using SFS resulted in more efficient integrations, whether or not the error estimate was modified. Using TR-BDF2 with SFS and **Est** resulted in a very accurate error estimate. For TR-BDF2 with SFS, it is unclear whether it is worthwhile to form **Est**. The matter is clearer with TRX2, which was integrated rather inefficiently with **est**, as evidenced by the maximum estimate error of 0.98. The maximum error in **est** observed when the modified estimate was used to control the integration was 40.34, and the average error was 14.11.

The analysis says that there is no advantage to modification when the problem is not stiff. This was confirmed by our integration of the van der Pol equation

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= \epsilon\left(1 - y_1^2\right)y_2 - y_1, \end{aligned}$$

Table 2
Problem D4

	Explicit first stage				Smoothed first stage			
	TR-BDF2		TRX2		TR-BDF2		TRX2	
	est	**Est**	**est**	**Est**	**est**	**Est**	**est**	**Est**
Successful steps	1041	26	2102	36	25	24	47	23
Error failures	642	0	990	0	0	0	2	0
Iteration failures	0	1	0	1	0	0	0	0
Function evaluations	7603	155	14776	217	78	75	185	114
Jacobian evaluations	1	2	1	2	1	1	1	1
LU decompositions	1619	20	2700	30	18	17	29	16
Linear system solves	6560	153	12672	215	76	97	183	135
Maximum estimate error	0.99	0.15	0.99	0.21	0.19	0.09	0.98	0.23
Average estimate error	0.37	0.04	0.40	0.07	0.07	0.02	0.35	0.06
Maximum local error	0.56	0.62	0.42	0.50	0.56	0.62	0.42	0.50
Average local error	0.01	0.13	0.03	0.15	0.11	0.12	0.04	0.09

Table 3
Nonstiff van der Pol problem

	Explicit first stage				Smoothed first stage			
	TR-BDF2		TRX2		TR-BDF2		TRX2	
	est	**Est**	**est**	**Est**	**est**	**Est**	**est**	**Est**
Successful steps	116	116	94	94	116	116	94	93
Error failures	22	23	21	21	22	24	20	19
Iteration failures	2	1	6	2	2	1	3	2
Function evaluations	658	669	580	586	529	557	480	482
Jacobian evaluations	3	2	7	3	3	2	7	3
LU decompositions	92	96	98	92	98	99	100	86
Linear system solves	540	690	484	605	527	695	478	592
Maximum estimate error	0.27	0.32	0.52	0.26	0.41	0.41	0.44	0.89
Average estimate error	0.07	0.07	0.11	0.07	0.08	0.07	0.11	0.11
Maximum local error	1.16	0.96	1.50	0.95	1.33	0.89	1.34	1.78
Average local error	0.38	0.38	0.36	0.36	0.38	0.37	0.35	0.36

over the interval $[0, 20]$ with initial values $(0, 0.25)^{\mathrm{T}}$ and parameter $\epsilon = 1$. The results are given in Table 3. There was no significant difference in the number of steps or function evaluations required, nor in the quality of the estimates.

The D4 problem is a scaled version of a problem of Robertson. Following [10], we solve in the original variables:

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3, \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2, \\ y_3' &= 3 \times 10^7 y_2^2. \end{aligned}$$

The equation is to be integrated over the interval $[0, 4 \times 10^7]$ with the initial conditions $(1, 0, 0)^{\mathrm{T}}$. The results are given in Table 4. This problem is quite stiff, and the TRX2 method simply could not handle it, nor could TR-BDF2 using EFS and **est**. Each of these unsuccessful runs was terminated after 50000 function evaluations. The run with TR-BDF2 using **Est** and EFS was suc-

Table 4
Robertson problem

	Explicit first stage				Smoothed first stage			
	TR-BDF2		TRX2		TR-BDF2		TRX2	
	est	**Est**	**est**	**Est**	**est**	**Est**	**est**	**Est**
Successful steps	x	631	x	x	84	76	x	x
Error failures	x	3	x	x	11	5	x	x
Iteration failures	x	901	x	x	9	10	x	x
Function evaluations	x	5560	x	x	462	399	x	x
Jacobian evaluations	x	288	x	x	9	10	x	x
LU decompositions	x	1512	x	x	79	77	x	x
Linear system solves	x	5561	x	x	460	478	x	x
Maximum estimate error	x	0.29	x	x	0.52	0.39	x	x
Average estimate error	x	0.01	x	x	0.14	0.14	x	x
Maximum local error	x	0.69	x	x	0.71	1.01	x	x
Average local error	x	0.02	x	x	0.31	0.41	x	x

cessful, but it was plagued with a large number of iteration failures and was very inefficient. It is important to appreciate that the reason TR-BDF2 has difficulty here is ultimately due to errors arising from not evaluating the formula exactly. Significant improvement can be realized by reducing κ and allowing more iterations, though the cost is still about twice that incurred with SFS. TRX2, on the other hand, simply lacks sufficient stability to be effective on this problem.

The results obtained with SFS were superior, and this time we see improvement with the modified error estimate. Some additional insight into the quality of the solution can be obtained by measuring how well the numerical solution satisfies the conservation law satisfied by the exact solution, $y_1 + y_2 + y_3 = 1$. The solution using SFS and **Est** satisfied the conservation law at each step with a maximum deviation of 1.55×10^{-15}, about 14 times the unit roundoff.

We are reluctant to compare our experimental code with any production-quality code, but it is useful to have a point of reference. The Robertson problem is the sample problem distributed with VODE [4], a production-quality variable order, variable coefficient BDF code, and it is interesting to note that given the same tolerances, VODE takes 400 steps, experiences 30 error failures, no iteration failures, 599 function evaluations, 9 Jacobian evaluations, 97 LU decompositions, and 598 linear system solves. At these tolerances the cost of using TR-BDF2 compares rather favorably.

8. Conclusion

TR-BDF2 implemented properly with the quality local error estimate and continuous extension given here is an attractive implicit one-step method. Indeed, these are the ingredients missing in the strongly S-stable methods thoroughly studied in [1]. In a simulation environment like SIMULINK [20] where restarts are common and linear algebra is unusually fast, it has a great deal to offer, especially since it is easy to combine the method in a single code with TRX2 for those simulations where damping at infinity is not appropriate [22, p. 304].

References

[1] R. Alexander, Diagonally implicit Runge–Kutta methods for stiff O.D.E.'s, *SIAM J. Numer. Anal.* 14 (1977) 1006–1021.

[2] R.E. Bank, W.M. Coughran Jr, W. Fichtner, E.H. Grosse, D.J. Rose and R.K. Smith, Transient simulation of silicon devices and circuits, *IEEE Trans. Comput.-Aided Design* 4 (1985) 436–451.

[3] M. Berzins and R.M. Furzeland, An adaptive theta method for the solution of stiff and nonstiff differential equations, *Appl. Numer. Math.* 9 (1992) 1–19.

[4] P.N. Brown, G.D. Byrne and A.C. Hindmarsh, VODE: a variable coefficient ODE solver, *SIAM J. Sci. Statist. Comput.* 10 (1989) 1038–1051.

[5] J. Carroll, A composite integration scheme for the numerical solution of systems of ordinary differential equations, *J. Comput. Appl. Math.* 25 (1989) 1–13.

[6] J. Carroll, A composite integration scheme for the numerical solution of systems of parabolic PDEs in one space dimension, *J. Comput. Appl. Math.* 46 (1993) 327–343.

[7] T.S. Chua and P.M. Dew, The simulation of a gas transmission network using a variable-step integrator, Rept. No. 148, Department of Computer Studies, University of Leeds, Leeds (1981).

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

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

[10] A.C. Hindmarsh and G.D. Byrne, Applications of EPISODE: an experimental package for the integration of ordinary differential equations, in: L. Lapidus and W.E. Schiesser, eds., *Numerical Methods for Differential Equations* (Academic Press, Orlando, FL, 1976) 147–166.

[11] M.E. Hosea, A new recurrence for computing Runge–Kutta truncation error coefficients, *SIAM J. Numer. Anal.* (to appear).

[12] N. Houbak, S.P. Nørsett and P.G. Thomsen, Displacement or residual test in the application of implicit methods for stiff problems, *IMA J. Numer. Anal.* 5 (1985) 297–305.

[13] S.P. Nørsett and P.G. Thomsen, Local error control in *SDIRK*-methods, *BIT* 26 (1986) 100–113.

[14] A. Prothero and A. Robinson, On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations, *Math. Comput.* 28 (1974) 145–162.

[15] R.E. Scraton, Some L-stable methods for stiff differential equations, *Internat. J. Comput. Math. B* 9 (1981) 81–87.

[16] L.F. Shampine, Implementation of implicit formulas for the solution of ODEs, *SIAM J. Sci. Statist. Comput.* 1 (1980) 103-118.

[17] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations* (Chapman & Hall, New York, 1994).

[18] L.F. Shampine and L.S. Baca, Error estimators for stiff differential equations, *J. Comput. Appl. Math.* 11 (1984) 197–207.

[19] L.F. Shampine and M.W. Reichelt, The MATLAB ODE suite, Rept. No. 94-6, Mathematics Department, Southern Methodist University, Dallas, TX (1994).

[20] SIMULINK 1.3, The Mathworks, Inc., Natick, MA (1995).

[21] T. Steihaug and A. Wolfbrandt, An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations, *Math. Comput.* 33 (1979) 521–534.

[22] A. Vladimirescu, *The SPICE Book* (Wiley, New York, 1994).

[23] H. Zedan, Modified Rosenbrock–Wanner methods for systems of stiff ordinary differential equations, Dissertation (1982); also Rept. CS-83-06: A preliminary assessment of the local error estimates of some Rosenbrock-type methods, School of Mathematics, Computer Science, University of Bristol, Bristol (1983).