MATHEMATICS OF COMPUTATION, VOLUME 33, NUMBER 146
APRIL 1979, PAGES 557–561

# On the Efficient Implementation of Implicit Runge-Kutta Methods

By J. M. Varah

Abstract. Extending some recent ideas of Butcher, we show how one can efficiently implement general implicit Runge-Kutta methods, including those based on Gaussian quadrature formulas which are particularly useful for stiff equations. With this implementation, it appears that these methods are more efficient than the recently proposed semiexplicit methods and their variants.

**I. Introduction.** Consider the initial value problem

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}), \qquad \mathbf{y}(a) = \mathbf{y}_0, \tag{1.1}$$

where $\mathbf{y}$ is an $m$-vector. Implicit Runge-Kutta methods for the numerical solution of (1.1), first proposed by Butcher [1964], have the form

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i F_i, \quad \text{where } F_i = f\left(y_n + h \sum_{j=1}^{s} a_{ij} F_j\right), \; i = 1, \ldots, s. \tag{1.2}$$

The methods are commonly described by the tableau

$$\begin{array}{c|c} a_{11} \cdots a_{1s} & c_1 \\ \vdots \quad\quad \vdots & \vdots \\ a_{s1} \cdots a_{ss} & c_s \\ \hline b_1 \cdots b_s & \end{array}$$

where $c_i = \sum_{j=1}^{s} a_{ij}$.

Butcher showed that these methods could have order as high as $2s$, and subsequently several authors have produced classes of formulas which are particularly appropriate for stiff equations. We mention the methods based on Gauss-Legendre quadrature (Butcher [1964]), those based on Radau quadrature (methods IA and IIA of Ehle [1969]), and those based on Lobatto quadrature (methods IIIC of Ehle [1969] and Chipman [1971]).

However, the implementation difficulties of these methods have precluded their general use; for each step, (1.2) involves solving a nonlinear algebraic system of order $ms$. Instead, other authors have proposed methods of this type with $A = (a_{ij})$ lower triangular to facilitate their numerical solution—these are the semiexplicit methods of

Received May 23, 1978.
*AMS (MOS) subject classifications* (1970). Primary 65L05.


0025-5718/79/0000-0056/$02.25

Norsett [1974]. Unfortunately, methods of this type have maximum order $s + 1$ (Norsett and Wolfbrandt [1977]) and making them useful for stiff equations restricts the possibilities even more, so that only a few such methods have been found (see Alexander [1977]).

The problem of efficient implementation has attracted a lot of attention recently; see for example Chipman [1973] and Bickhart [1977]. As well, Butcher [1976] has described an ingenious technique for implementing general implicit Runge-Kutta methods using a similarity transformation $T^{-1}AT = B$, where $B$ has a much simpler structure. In Butcher [1977], he then applies his technique to a class of special methods of Burrage [1977], which have order $s$ or $s + 1$, and for which the matrix $B$ is a single Jordan block.

In this paper, we would like to point out how this technique can be efficiently applied to general implicit Runge-Kutta methods, and thus (we hope) render the Gauss-Legendre, Gauss-Radau, the Gauss-Lobatto formulas more effective and competitive. The key is to perform a similarity transformation to Hessenberg form on the Jacobian matrix, rather than use the $LU$ factorization. We also make a rough estimate of the work involved per step of the relevant methods.

**II. Using the Similarity Transformation.** Each step of (1.2) requires the solution of the nonlinear system

$$\phi(F) = 0,$$

where $\phi(F) = (\phi_1(F), \ldots, \phi_s(F))^T$ and $\phi_i(F) = F_i - f(y_n + h\Sigma a_{ij}F_j)$. Normally, this is done by a modified form of Newton's method, one step of which takes $F$ into $F + \Delta F$, with $\Delta F$ given by

$$\tilde{J}(\Delta F) = -\phi(F). \tag{2.1}$$

Here

$$\tilde{J} = \begin{pmatrix} I - ha_{11}J & -ha_{12}J & \cdots & \\ -ha_{21}J & I - ha_{22}J & & \vdots \\ \vdots & & & \\ \cdots & \cdots & \cdots & I - ha_{ss}J \end{pmatrix} = I - h(A \otimes J),$$

where $J$ is the Jacobian matrix, $J_{ij} = \partial f_i/\partial y_j$, normally evaluated at most once per step.

If we use the similarity transformation $T^{-1}AT = B$,

$$\tilde{J} = I - h(A \otimes J) = I - h(T \otimes I)(B \otimes J)(T^{-1} \otimes I)$$

so (2.1) reduces to

$$(I - h(B \otimes J))(\overline{\Delta F}) = -\overline{\phi(F)}$$

in the transformed coordinate system where $\bar{X} = (T^{-1} \otimes I)X$, for any vector $X$. Thus, keeping in the transformed coordinate system as much as possible, one step of Newton's method can be described as follows:

(A) given $\overline{F}$, compute $-\overline{\phi_i(F)} = \overline{f\left(y_n + h\sum a_{ij}F_j\right)} - \overline{F}_i, \quad i = 1, \ldots, s,$

(B) solve $(I - h(B \otimes J))(\overline{\Delta F}) = -\overline{\phi(F)}$, and form $\overline{F} + \overline{\Delta F}$.

Thus, we can avoid explicitly calculating $F$; we iterate for a fixed number of steps or until $\|\overline{\phi(F)}\|$ is small, and finally form

$$y_{n+1} = y_n + h\sum b_j F_j$$

via

$$y_{n+1} = y_n + h\sum \overline{b}_j \overline{F}_j, \tag{2.3}$$

where $\overline{\mathbf{b}} = T^T\mathbf{b}$.

We now describe the work involved in (A) and (B). In (A), to get the new arguments $z_i = y_n + h\Sigma a_{ij}F_j$, $i = 1, \ldots, s$, we need to compute

$$\mathbf{Z} = y_n \otimes I + h(A \otimes I)F = y_n \otimes I + h(TB \otimes I)\overline{F},$$

evaluate $f(z_i)$, $i = 1, \ldots, s$, and compute $\overline{f(Z)} = (T^{-1} \otimes I)f(Z)$. Each transformation takes $ms^2$ multiplications, which we use as our basic measure of time; thus, the total is $(2ms^2 + sm\hat{f})$ where $\hat{f}$ denotes the equivalent number of multiplications needed to evaluate one component of the function $f(z)$.

This first step (A) is the same for all methods; (B), however, will depend on the particular method used. For the Butcher/Burrage scheme mentioned earlier, which uses Laguerre polynomials and has order $s$ or $s + 1$,

$$B = \begin{pmatrix} \lambda & & \\ -\lambda & \lambda & \\ & \ddots & \ddots \\ & -\lambda & \lambda \end{pmatrix}, \quad \text{so } I - h(B \otimes J) = \begin{pmatrix} I - h\lambda J & & \\ h\lambda J & I - h\lambda J & \\ & \ddots & \ddots \\ & h\lambda J & I - h\lambda J \end{pmatrix}.$$

This is lower block-triangular, so we can use a forward block recurrence to solve it. Expressed efficiently (as in Butcher [1977]), the recurrence is

$$(I - h\lambda J)(\overline{\Delta F})_1 = -\overline{\phi(F)}_1,$$

$$(I - h\lambda J)((\overline{\Delta F})_i - (\overline{\Delta F})_{i-1}) = -\overline{\phi(F)}_i - (\overline{\Delta F})_{i-1}, \quad i = 2, \ldots, s.$$

The work involved here is one $LU$ factorization of $(I - h\lambda J)$, plus the solution of $s$ right-hand sides for each Newton iteration. Thus, if we assume $q$ Newton iterations are performed, the total work (in multiplications) per step is

$$w_L = m^2\hat{J} + \frac{m^3}{3} + q(2ms^2 + m^2 s + sm\hat{f}). \tag{2.4}$$

Here $\hat{J}$ is the equivalent number of multiplications to evaluate *one* component of the

Jacobian. We hasten to add that these estimates only involve the highest order terms. For example, other $ms$ multiplications are needed to form $y_{n+1}$ from (2.3).

A second way of coping with the problem, which applies to any implicit Runge-Kutta method, is to effect a transformation to diagonal form:

$$T^{-1}AT = B = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_s \end{pmatrix}.$$

For the useful methods (like Gauss-Legendre), the eigenvalues are complex, so the operations involved are complex. If programmed directly, a complex multiplication involves four real ones, but in a language with complex type declarations it may be much less. In Fortran on IBM 370 machines for example, a factor of two is more realistic.

Thus, in (B), $(I - h(B \otimes J))$ is block-diagonal, with the $i$th block $(I - h\lambda_i J)$. If the usual $LU$ decomposition is used on each block, it would involve $sm^3/3$ multiplications, which would be very expensive. Instead, one can use a triangular similarity transformation to Hessenberg form on the matrix $J$: $LJL^{-1} = H$. This involves $5m^3/6$ multiplications and need only be done once. This is used as a basic step in solution of the general eigenproblem (see Chapter 7 of Wilkinson [1965]) and has also been suggested in connection with multistep methods by Enright [1976].

Using this transformation, $I - h\lambda_i J = L^{-1}(I - h\lambda_i H)L$, so the systems we must solve,

$$(I - h\lambda_i J)(\overline{\Delta F})_i = -\overline{\phi(F)}_i,$$

become

$$(I - h\lambda_i H)(L(\overline{\Delta F})_i) = -L(\overline{\phi(F)})_i, \tag{2.5}$$

or in product form

$$(I - h(B \otimes H))(\overline{\overline{\Delta F}}) = -\overline{\overline{\phi(F)}},$$

where $\overline{\overline{X}} = (I \otimes L)\overline{X}$. The transformations $\overline{\phi(F)} \longrightarrow \overline{\overline{\phi(F)}}$ and $\overline{\overline{\Delta F}} \longrightarrow \overline{\Delta F}$ involve $s$ multiplications by $L$ or linear system solutions with $L$; each of these requires $sm^2/2$ complex multiplications. And, since $H$ is Hessenberg, (2.5) requires only $sm^2$ multiplications. Thus, if we again assume $q$ Newton iterations are performed, the total work (in multiplications) per step is

$$w_G = m^2\hat{J} + \frac{5}{6}m^3 + q(2ms^2c + 2sm^2c + sm\hat{f}). \tag{2.6}$$

Here $c$ denotes the effective factor to reflect complex multiplications.

A third alternative for those schemes derived from Gaussian quadrature formulas, which keeps operations in the real domain, is to transform $A$ into tridiagonal form $B$. This uses a $T$ whose values are obtained from the relevant orthogonal polynomials, much as in Butcher [1977] for the Laguerre polynomials. Then $I - h(B \otimes J)$ is a real block-tridiagonal matrix, and one can proceed with a block-tridiagonal $LU$ factorization. Unfortunately the Hessenberg structure is not maintained, so it seems impossible to avoid doing $O(sm^3)$ multiplications, making this too costly.

**III. Comparison of Methods.** It is difficult to give meaningful comparisons of a quantitative nature on the basis of rough estimates like (2.4) and (2.6). However, since the Butcher/Burrage methods have order $s$ or $s + 1$, whereas the Gauss methods have order $2s$ or $2s - 1$, it seems clear that the Gauss methods require less work per step for the same order method. As well, it appears as though the error constants for the Butcher/Burrage methods are larger, especially for $A$-stable methods, so more steps would be required for the same accuracy.

It is even more difficult to compare these methods with stiff multistep methods like those of Gear. A direct estimate of work per step for an $r$-step method gives

$$w_{MS} = m^2\hat{J} + \frac{m^3}{3} + rm + q(m^2 + m\hat{f}).$$

These methods would seem to have a clear advantage over either type of Runge-Kutta scheme of the same order. However, this only applies to the basic implicit multistep schemes which have order $\leqslant 6$. Higher order multistep methods have been proposed (see for example Enright [1973], Varah [1978]) but they involve more work. In any case, because of the intricacies of error estimation and stepsize control which must be included in any useful code, it is much more meaningful to compare methods empirically on a suitable set of test problems.

Department of Computer Science
The University of British Columbia
Vancouver, B. C. V6T 1W5, Canada

R. ALEXANDER [1977], "Diagonally implicit Runge-Kutta methods for stiff ODE's," *SIAM J. Numer. Anal.*, v. 14, pp. 1006–1021.

KEVIN BURRAGE [1977], *A Special Family of Runge-Kutta Methods for Solving Stiff Differential Equations,* Tech. Rep. 122, Math. Dept., University of Auckland.

T. A. BICKHART [1977], "An efficient solution process for implicit Runge-Kutta methods," *SIAM J. Numer. Anal.*, v. 14, pp. 1022–1027.

J. C. BUTCHER [1964], "Implicit Runge-Kutta processes," *Math. Comp.*, v. 18, pp. 50–64.

J. C. BUTCHER [1976], "On the implementation of implicit Runge-Kutta methods," *BIT,* v. 16, pp. 237–240.

J. C. BUTCHER [1977], *A Transformed Implicit Runge-Kutta Method*, Tech. Rep. 111, Math. Dept., University of Auckland.

F. H. CHIPMAN [1971], "$A$-stable Runge-Kutta processes," *BIT,* v. 11, pp. 384–388.

F. H. CHIPMAN [1973], "The implementation of Runge-Kutta implicit processes," *BIT,* v. 13, pp. 391–393.

B. L. EHLE [1969], *On Padé Approximations to the Exponential Function and A-Stable Methods for the Numerical Solution of Initial Value Problems,* Res. Rep. CSRR 2010, Computer Science Dept., University of Waterloo.

W. H. ENRIGHT [1973], "Second derivative multistep methods for stiff ordinary differential equations," *SIAM J. Numer. Anal.*, v. 11, pp. 321–331.

W. H. ENRIGHT [1976], *Improving the Efficiency of Matrix Operations in the Numerical Solution of Stiff ODE's,* Tech. Rep. 98, Computer Science Dept., Univ. of Toronto.

S. P. NORSETT [1974], *Semi-Explicit Runge-Kutta Methods,* Report 6/74, Math. Dept., University of Trondheim, Norway.

S. P. NORSETT & A. WOLFBRANDT [1977], "Attainable order of rational approximations to the exponential function with only real poles," *BIT*, v. 17, pp. 200–208.

J. M. VARAH [1978], "Stiffly stable linear multistep methods of extended order," *SIAM J. Numer. Anal.*, v. 15, pp. 1234–1246.

J. H. WILKINSON [1965], *The Algebraic Eigenvalue Problem,* Clarendon Press, Oxford.