SIAM J. NUMER. ANAL.
Vol. 14, No. 6, December 1977

AN EFFICIENT SOLUTION PROCESS FOR IMPLICIT RUNGE–KUTTA METHODS*

THEODORE A. BICKART†

Abstract. Described in this paper is an efficient solution procedure for a modified Newton method solution of a ν-stage implicit Runge–Kutta method applied to a set of n first order ordinary differential equations. The procedure requires only order νn^3 scalar multiplications for each time-step plus order $\nu^2 n^2$ scalar multiplications for each iteration-step within the time-step. Also, it only requires space to store order $3n^3$ variables.

1. Introduction. High order, A-stable numerical integration can be achieved with implicit Runge–Kutta methods [7, pp. 243–244]. However, there is an impediment to using an implicit Runge–Kutta method. It is that a ν-stage method used to solve a system of n first order ordinary differential equations requires at each time-step the solution of νn nonlinear algebraic equations. If these algebraic equations are solved by the Newton method, or one of the variations on it, it is necessary to solve νn linear algebraic equations at each iteration-step within each time-step. Unless the system of equations is sparse and sparse matrix techniques are invoked, each iteration-step requires order $\nu^3 n^3$ scalar multiplications. We shall describe, for a particular modified Newton method, a solution procedure which requires only order νn^3 scalar multiplications for each time-step plus order $\nu^2 n^2$ scalar multiplications for each iteration-step within the time-step. Note: The procedure to be described is similar to that which has been employed with composite multistep methods, as alluded to in [1] and described in [8].

2. Solution process. Consider the n-vector-valued first order differential equation

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t) \tag{1}$$

and the initial condition

$$\mathbf{y}(t_0) = \mathbf{y}_0. \tag{2}$$

Let $\mathbf{y}_m$ denote the approximation of $\mathbf{y}(t_m)$, where $t_m = t_0 + mh$, derived by the following Runge–Kutta method:

$$\mathbf{y}_m = \mathbf{y}_{m-1} + h \sum_{i=1}^{\nu} \gamma_i \mathbf{k}_i, \tag{3}$$

where

$$\mathbf{k}_i = \mathbf{f}(\mathbf{y}_{m-1} + h \sum_{j=1}^{\nu} \beta_{ij} \mathbf{k}_j, t_{m-1} + \delta_i h) \qquad (i = 1, \cdots, \nu). \tag{4}$$

* Received by the editors February 26, 1976, and in final revised form November 19, 1976.

† Department of Electrical and Computer Engineering, Syracuse University, Syracuse, New York. On leave from Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, California 94720.

Now, set

$$\mathbf{p}_i = \mathbf{y}_{m-1} + h \sum_{j=1}^{\nu} \beta_{ij}\mathbf{k}_j \qquad (i = 1, \cdots, \nu), \tag{5}$$

or, equivalently,

$$\mathbf{p}_i = \mathbf{y}_{m-1} + h \sum_{j=1}^{\nu} \beta_{ij}\mathbf{f}_j \qquad (i = 1, \cdots, \nu), \tag{6}$$

where

$$\mathbf{f}_j = \mathbf{f}(\mathbf{p}_j, t_{m-1} + \delta_j h). \tag{7}$$

Note: $\mathbf{p}_i$ may be viewed as an approximation of $\mathbf{y}(t)$ for $t = t_{m-1} + \delta_i h \in [t_{m-1}, t_m]$. Suppose the matrix $[\beta_{ij}]$ is nonsingular;[1] then (6) is equivalent to

$$h\mathbf{f}_i = \sum_{j=1}^{\nu} \alpha_{ij}\mathbf{p}_j - \left[\sum_{j=1}^{\nu} \alpha_{ij}\right]\mathbf{y}_{m-1} \qquad (i = 1, \cdots, \nu), \tag{8}$$

where α_{ij} is the (i, j)th element of $[\beta_{ij}]^{-1}$. The set of equations (8) can be expressed in matrix form as

$$(\mathbf{A} \otimes \mathbf{U}_n)\hat{\mathbf{P}} - h\hat{\mathbf{D}} - (\mathbf{a} \otimes \mathbf{U}_n)\mathbf{y}_{m-1} = \mathbf{0}, \tag{9}$$

where $\otimes$ denotes the Kronecker product, $\mathbf{U}_n$ is a unit matrix of order n, $\mathbf{A} = [\alpha_{ij}]$,

$$\mathbf{a} = \begin{bmatrix} \sum_{j=1}^{\nu} \alpha_{ij} \\ \vdots \\ \sum_{j=1}^{\nu} \alpha_{\nu j} \end{bmatrix}, \quad \hat{\mathbf{P}} = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_\nu \end{bmatrix}, \quad \text{and} \quad \hat{\mathbf{D}} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_\nu \end{bmatrix}.$$

The $(l+1)$st Newton iterate for $\hat{\mathbf{P}}$, namely $\hat{\mathbf{P}}^{l+1}$, can be obtained from the solution $(\hat{\mathbf{P}}^{l+1} - \hat{\mathbf{P}}^l)$ of

$$\{(\mathbf{A} \otimes \mathbf{U}_n) - h\mathbf{J}^l\}(\hat{\mathbf{P}}^{l+1} - \hat{\mathbf{P}}^l) = -\{(\mathbf{A} \otimes \mathbf{U}_n)\hat{\mathbf{P}}^l - h\hat{\mathbf{D}}^l - (\mathbf{a} \otimes \mathbf{U}_n)\mathbf{y}_{m-1}\}, \tag{10}$$

where $\mathbf{J}^l$ is the block diagonal matrix

$$\mathbf{J}^l = \operatorname{diag}\{\mathbf{f}_{\mathbf{y}}(\mathbf{p}_1^l, t_{m-1} + \delta_1 h), \cdots, \mathbf{f}_{\mathbf{y}}(\mathbf{p}_\nu^l, t_{m-1} + \delta_\nu h)\}. \tag{11}$$

Note: Under the correspondence expressed in (5), the Newton iteration applied to (9) for the $\mathbf{p}_i$—elements of $\hat{\mathbf{P}}$—is equivalent to a Newton iteration applied to (4) for the $\mathbf{k}_i$.

Let the modified Newton method be that obtained by replacing $\mathbf{J}^l$ by a block diagonal matrix $\mathbf{J}$ of Jacobians of $\mathbf{f}$ evaluated not at the various intermediate points in the time-interval $[t_{m-1}, t_m]$ at each iteration-step l but at one point $t_{\hat{m}}$, where $\hat{m} \in [0, m-1]$. That is,

$$\mathbf{J} = \operatorname{diag}\{\mathbf{F}_{\hat{m}}, \cdots, \mathbf{F}_{\hat{m}}\} = \mathbf{U}_\nu \otimes \mathbf{F}_{\hat{m}}, \tag{12}$$

[1] It is easily shown using Butcher's results [2] that his order 2ν methods, which were shown to be A-stable by Ehle [5], satisfy this assumption. Thus, the set of methods for which the assumption is valid is not only nonvacuous but contains methods of some significance.

where $\mathbf{U}_\nu$ is a unit matrix of order ν and

$$\mathbf{F}_{\hat{m}} = \mathbf{f}_{\mathbf{y}}(\mathbf{y}_{\hat{m}}, t_{\hat{m}}). \tag{13}$$

Using (12) the modified Newton method that follows from (10) is

$$\{(\mathbf{A}\otimes\mathbf{U}_n) - h(\mathbf{U}_\nu\otimes\mathbf{F}_{\hat{m}})\}(\hat{\mathbf{P}}^{l+1}-\hat{\mathbf{P}}^l) = -\{(\mathbf{A}\otimes\mathbf{U}_n)\hat{\mathbf{P}}^l - h\hat{\mathbf{D}}^l - (\mathbf{a}\otimes\mathbf{U}_n)\mathbf{y}_{m-1}\}. \tag{14}$$

This is equivalent to the two-sided matrix equation

$$(h\mathbf{F}_{\hat{m}})(\mathbf{P}^{l+1}-\mathbf{P}^l) - (\mathbf{P}^{l+1}-\mathbf{P}^l)\mathbf{A}' = \mathbf{E}^l, \tag{15}$$

where $\mathbf{P}^l$, and $\mathbf{E}^l$, and $\mathbf{D}^l$ are the $n\times\nu$ matrices

$$\mathbf{P}^l = [\mathbf{p}_1^l, \cdots, \mathbf{p}_\nu^l], \qquad \mathbf{E}^l = \mathbf{P}^l\mathbf{A}' - h\mathbf{D}^l - \mathbf{y}_{m-1}\mathbf{a}',$$

and

$$\mathbf{D}^l = [\mathbf{f}_1^l, \cdots, \mathbf{f}_\nu^l],$$

with $\mathbf{A}'$ being, of course, the transpose of $\mathbf{A}$.

Let $a(\lambda)$ denote the characteristic polynomial of $\mathbf{A}'$; that is, let

$$a(\lambda) = \det\{\lambda\mathbf{U}_\nu - \mathbf{A}'\} = \sum_{k=0}^{\nu} a_k\lambda^{\nu-k}. \tag{16}$$

Then, as Jameson has shown [6], the solution of (15) can be expressed as

$$\mathbf{P}^{l+1}-\mathbf{P}^l = \{a(h\mathbf{F}_{\hat{m}})\}^{-1}\left\{\sum_{k=0}^{\nu-1} a_k\mathbf{M}_{\nu-k}\right\}, \tag{17}$$

where

$$\mathbf{M}_0 = \mathbf{0}, \quad \mathbf{M}_1 = \mathbf{E}^l, \quad \mathbf{M}_k = (h\mathbf{F}_{\hat{m}})\mathbf{M}_{k-1} + \mathbf{M}_{k-1}\mathbf{A}' - (h\mathbf{F}_{\hat{m}})\mathbf{M}_{k-2}\mathbf{A}'.$$

Note that if $m-1 > \hat{m}$, then $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$ was evaluated at a previous time-step and need not be evaluated at the current time-step. On the other hand, if $m-1=\hat{m}$, then $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$ must be evaluated; this entails 1 Jacobian matrix evaluation and $\nu n^3 + n^2$ scalar multiplications. The number of scalar multiplications,[2] vector function evaluations, and Jacobian matrix evaluations are displayed in Table 1. The number of iteration-steps at the mth time-step is there denoted l_m.

TABLE 1
Scalar multiplications

$m-1$	New method (Solution for $\mathbf{P}$)	Old method (Solution for $\hat{\mathbf{P}}$)	Vector function evaluations	Jacobian matrix evaluations
$>\hat{m}$	$\nu n + \nu$ $+ l_m\{\nu^2n^2 + (\nu^3+\nu^2)n\}$	$\nu n + \nu$ $+ l_m\{\nu^2n^2 + (\nu^2+\nu)n\}$	$l_m\nu$	0
$=\hat{m}$	$\nu n^3 + n^2 + \nu n + \nu$ $+ l_m\{\nu^2n^2 + (\nu^3+\nu^2)n\}$	$\nu^3n^3 + n^2 + \nu n + \nu$ $+ l_m\{\nu^2n^2 + (\nu^2+\nu)n\}$	$l_m\nu$	1

[2] Counts of multiplications include the number of divisions.

Shown also in Table 1, for purposes of comparison, are the number of multiplications needed when $\hat{\mathbf{P}}^l$ is evaluated by solving (14) using an explicit inverse of $\{(\mathbf{A}\otimes\mathbf{U}_n)-h(\mathbf{U}_\nu\otimes\mathbf{F}_{\hat{m}})\}$. (The numbers of vector function evaluations and Jacobian matrix evaluations are the same for both methods.)

For large n and $m-1=\hat{m}$—the worst case—the dominant terms in the expression for the number of scalar multiplications in a time-step are $\nu n^3+l_m\nu^2n^2$, as was predicated in the introduction. For purposes of comparison, the dominant terms when solving (14) directly are $\nu^3n^3+l_m\nu^2n^2$. Note: If LU decomposition rather than explicit inversion is used, the dominant terms become, respectively, $(\nu-2/3)n^3+l_m\nu^2n^2$ and $(1/3)\nu^3n^3+l_m\nu^2n^2$. *Remark*: The value of l_m to achieve a solution has been observed to be essentially the same for both methods—old and new.

Another factor of some significance in evaluating the new method for determining the $\mathbf{p}_i$ is the amount of data storage space required. The number of variables for which space must exist when using an explicit inverse is shown in Table 2 for the new and old methods. There, ε_n and ε_o denote small amounts of storage space needed during intermediate calculations. Observe, for large n, that the dominant terms are $3n^2$ and $(2\nu^2+1)n^2$ for the new and old methods, respectively. (If LU decomposition rather than explicit inversion is used, then these dominant terms become, respectively, $2n^2$ and $(\nu^2+1)n^2$.) Thus, the dominant number of variables as well as the dominant number of multiplications, is smaller for the new method by a multiplicative factor of order $1/\nu^2$.

Upon convergence of the iteration—l_m iteration steps—$\mathbf{P}$ is assigned the value of $\mathbf{P}^{l_m}$. From the columns of $\mathbf{P}$, which are the vectors $\mathbf{p}_j$ $(j=1,\cdots,\nu)$, the value of $\mathbf{y}_m$ can be evaluated using

$$\mathbf{y}_m=\sigma_0\mathbf{y}_{m-1}+\sum_{j=1}^{\nu}\sigma_j\mathbf{p}_j, \tag{18}$$

an expression derived by combining (3) through (8), excluding (6), with

$$\sigma_j=\sum_{i=1}^{\nu}\gamma_i\alpha_{ij}\qquad(j=1,\cdots,\nu)$$

and

$$\sigma_0=1-\sum_{j=1}^{\nu}\sigma_j.$$

TABLE 2

Number of stored variables

New method (Solution for $\mathbf{P}$)	Old method (Solution for $\hat{\mathbf{P}}$)
$3n^2$ $+(7\nu+1)n+(\nu^2+4\nu+4)$ $+\varepsilon_n$	$(2\nu^2+1)n^2$ $+(4\nu+1)n+(\nu^2+3\nu+4)$ $+\varepsilon_o$

3. Discussion. The improvement based on operations and storage space counts is rendered somewhat more real by the following, albeit limited, numerical data. This data is based on realization of the methods as APL functions executed on an IBM System 370 Model 155 computer. For $\nu = 3$ it was found that n could be as large as 29 for the new method, versus 12 for the old method, in a fixed size APL workspace. Note: For storage space, the dominant terms would indicate a need for similar amounts of space—2523 stored variables for the new method, versus 2736 for the old method. For $\nu = 3$, $n = 12$, $l_m = 1$, and $m - 1 = \hat{m}$ it was found that the old method required about 5 times as much central processor time as the new method, for other than vector function and Jacobian matrix evaluations. Note: For operations, the dominant terms would indicate a need for 7.4 times as many multiplications by the old method.

The new method engenders a quite significant reduction in the number of multiplications because the matrix to be inverted is of order $n \times n$, not $\nu n \times \nu n$. Chipman has also proposed a method—alluded to in [3] and described in [4]—which only requires inversion of an order $n \times n$ matrix; however, not only is his method different, it applies to a restricted class of implicit Runge–Kutta methods. *Added in proof*: See also [9], a new result, by Butcher.

Implicit Runge–Kutta methods, as previously footnoted, can manifest both high order and A-stability—properties associated with an efficient numerical solution of a stiff ordinary differential equation. Since for such an equation the condition number of $h\mathbf{F}_{\hat{m}}$ tends to be rather large, it can be expected that the conditon number of $a(h\mathbf{F}_{\hat{m}})$ increases—significantly for sufficiently large h—as the degree ν of $a(\lambda)$ increases. On occasion $a(h\mathbf{F}_{\hat{m}})$ may be too ill-conditioned to invert using fixed precision arithmetic. If this should be the case, the inverse of $a(h\mathbf{F}_{\hat{m}})$ can be sought in the following manner. Let $a(\lambda)$ be expressed as a product (of not necessarily irreducible) real polynomials; thus,

$$a(\lambda) = \prod_{i=1}^{\mu} c_i(\lambda), \tag{19}$$

where $\mu \leqq \nu$. Then,

$$\{a(h\mathbf{F}_{\hat{m}})\}^{-1} = \prod_{i=1}^{\mu} \{c_i(h\mathbf{F}_{\hat{m}})\}^{-1}. \tag{20}$$

The task is to select the factors $c_i(\lambda)$ such that each $c_i(h\mathbf{F}_{\hat{m}})$ can be inverted. Ideally, μ should also be as small as possible, since, evaluated in this way, $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$ requires $(\mu - 1)n^3$ additional scalar multiplications for a total of order $(\nu + \mu - 1)n^3 + l_m\nu^2n^2$ scalar multiplications when $m - 1 = \hat{m}$. However, achievement of a least μ need not be of paramount concern, because, as $\mu \leqq \nu$, the total cannot exceed $(2\nu - 1)n^3 + l_m\nu^2n^2$, a value still less—often significantly so—than the $\nu^3n^3 + l_m\nu^2n^2$ of the old method. It is of some note that this variation on the new method requires but little additional space for stored variables—just n^2 additional variables must be stored.

The factored form of $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$, which can of course be used even if $a(h\mathbf{F}_{\hat{m}})$ is not ill-conditioned, suggests yet another variation on the new method. This variation will achieve the condition improvement of the factored form with (most

likely) fewer additional multiplications but at a not necessarily insignificant increase in required storage space—$(\mu-1)n^2$ additional stored variables. Now, if, rather than first evaluating $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$ as in (20) for use in (17), the expression (17) using (20) is rewritten as

$$\mathbf{P}^{l+1}-\mathbf{P}^{l}=\{c_1(h\mathbf{F}_{\hat{m}})\}^{-1}\cdots\{c_\mu(h\mathbf{F}_{\hat{m}})\}^{-1}\left\{\sum_{k=0}^{\nu-1}a_k\mathbf{M}_{\nu-k}\right\} \tag{21}$$

and the several matrix products are evaluated right-to-left at each iteration-step, then just $l_m(\mu-1)\nu n^2$ additional scalar multiplications are needed for a total of order $\nu n^3+l_m[\nu^2+(\mu-1)\nu]n^2$ scalar multiplications when $m-1=\hat{m}$. As $\mu\leqq\nu$, this total cannot exceed $\nu n^3+l_m(2\nu^2-\nu)n^2$, a number not much greater than when $\{a(h\mathbf{F}_{\hat{m}})\}^{-1}$ is evaluated by the expression (17). If LU decomposition rather than explicit inversion is used with (21), then the number of scalar multiplications will be of order $[\nu-(2/3)\mu]n^3+l_m[\nu^2+(\mu-1)\nu]n^2$. Though this is a quantity which could get rather small—between $(1/3)\nu n^3+l_m[2\nu^2-\nu]n^2$ and $[(2/3)\nu-(1/3)]n^3+l_m[(3/2)\nu^2-\nu/2]n^2$ or $(2/3)\nu n^3+l_m[(3/2)\nu^2-\nu]n^2$, depending upon the number of quadratic factors in an irreducible factorization of $a(\lambda)$—the larger storage space requirement might mitigate its use.

A comment on the applicability of sparse matrix techniques in conjunction with the new method appears to be an appropriate way to close this discussion. Many ordinary differential equations, typically when n is very large, exhibit a sparse Jacobian matrix, $\mathbf{F}_{\hat{m}}$. Because of the powers of that matrix, it can be assumed that $a(h\mathbf{F}_{\hat{m}})$ is less sparse. However, if $\mathbf{F}_{\hat{m}}$ is very sparse—somewhat likely when n is very large—and if the degree ν of $a(\lambda)$ is not too great, then $a(h\mathbf{F}_{\hat{m}})$ may remain sufficiently sparse to take advantage of sparse matrix techniques. That would be the case, for example, if $\mathbf{F}_{\hat{m}}$ were a band matrix with a sufficiently small bandwidth. Note also that applicability of sparse matrix techniques—in particular, sparse storage—might be the determining factor in using (21), possibly with LU decomposition, rather than (17).

REFERENCES

[1] T. A. BICKART AND Z. PICEL, *High order stiffly stable composite multistep methods for numerical integration of stiff differential equations*, BIT, 13 (1973), pp. 272–286.

[2] J. C. BUTCHER, *Implicit Runge–Kutta processes*, Math. Comp., 18 (1964), pp. 50–64.

[3] F. H. CHIPMAN, *The implementation of Runge–Kutta processes*, BIT, 13 (1973), pp. 391–393.

[4] ———, *Numerical solution of initial value problems using A-stable Runge–Kutta processes*, Univ. of Waterloo Dept. of Applied Analysis and Computer Science Res. Rep. CSRR 2042, Waterloo, Ontario, Canada, June 1971.

[5] B. L. EHLE, *High order A-stable methods for the numerical solution of systems of D.E.'s*, BIT, 8 (1968), pp. 276–278.

[6] A. JAMESON, *Solution of the equation $AX+XB=C$ by inversion of an $M\times M$ or $N\times N$ matrix*, SIAM J. Appl. Math., 16 (1968), pp. 1020–1023.

[7] J. D. LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley, New York, 1973.

[8] Z. PICEL, *Block one-step methods with free parameters for numerical integration of stiff ordinary differential equations*, Ph.D. Dissertation, Syracuse Univ., Syracuse, New York, September 1975.

[9] J. C. BUTCHER, *On the implementation of implicit Runge–Kutta methods*, BIT, 16 (1976), pp. 237–240.