MATHEMATICS OF COMPUTATION
VOLUME 40, NUMBER 161
JANUARY 1983, PAGES 175–191

Block Runge-Kutta Methods for the Numerical Integration of Initial Value Problems in Ordinary Differential Equations Part I. The Nonstiff Case

By J. R. Cash

Abstract. Block Runge-Kutta formulae suitable for the approximate numerical integration of initial value problems for first order systems of ordinary differential equations are derived. Considered in detail are the problems of varying both order and stepsize automatically. This leads to a class of variable order block explicit Runge-Kutta formulae for the integration of nonstiff problems and a class of variable order block implicit formulae suitable for stiff problems. The central idea is similar to one due to C. W. Gear in developing Runge-Kutta starters for linear multistep methods. Some numerical results are given to illustrate the algorithms developed for both the stiff and nonstiff cases and comparisons with standard Runge-Kutta methods are made.

1. Introduction. In the first part of this paper we will be concerned with the approximate numerical integration of the nonstiff initial value problem

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0, y \in \mathbf{R}^s. \tag{1.1}$$

In [1], Bond presented a family of block cyclic schemes suitable for the numerical integration of (1.1). The approach developed by Bond can be regarded as a cyclic approach [6], a deferred correction approach [9], a linear multistep method with an off-step point [2], [3], [12], [16] or a block approach [19], [22], [28]. However, a rather more fruitful interpretation is to regard these formulae as explicit Runge-Kutta methods, and the purpose of this paper is to extend Bond's formulae in a Runge-Kutta framework using Butcher's analysis.

Our aim will be to derive block formulae of order p which integrate p steps forward and which contain built-in local error estimates at each step. An alternative way of looking at this is to regard our formulae as being pth order explicit Runge-Kutta methods integrating forward over a single step $H \equiv ph$ and yielding $O(H/p)^{p+1}$ accurate solutions at $p - 1$ equally spaced internal nodes. This problem has been investigated by Iserles [17] for fully implicit Runge-Kutta formulae, and he develops what he calls B and E classes of formulae. The main difference between Iserles' approach and the approach we consider is firstly that we will only be concerned with explicit and diagonally implicit Runge-Kutta formulae, and secondly our formulae will be such that an estimate of the local truncation error is available at all internal nodes. Block explicit Runge-Kutta formulae have also been considered

Received December 30, 1980; revised February 2, 1982 and June 21, 1982.
1980 *Mathematics Subject Classification.* Primary 65L05.

0025-5718/82/0000-0710/$04.75

by Rosser [19] and by Sarafyan [20], [21]. However, the formulae which they present have lower orders of accuracy at internal points of the block than at the end of the block. In contrast to this, our formulae will be designed so that $O(h^{p+1})$ accurate approximations are obtained at all nodes, and we will explain later why we make this restriction. Finally we mention that the central idea in this paper is very similar to one due to Gear [11]. In [11] Runge-Kutta methods are proposed that yield $O(h^{p+1})$ accurate approximations to $h^s y^{(s)}(x_n)$ for $s = 1, 2, \ldots, p$. These formulae are not used to carry out the integration but instead generate the additional information required to start linear multistep methods at high order. However, many of the ideas given in [11] carry over to our approach, and the present paper should be regarded as complementary to the paper by Gear.

One of the main reasons for considering the block formulae introduced in this paper is that they allow us to change order easily and, perhaps more importantly, they allow us to choose the best order ($\leqslant 4$) to use when starting. In an extensive survey by Hull et al. [14], [15] it was found, for the test problems considered, that a fourth order Runge-Kutta formula performs well when function evaluations are simple and the imposed tolerance is not very stringent. For very strict tolerances an eighth order formula due to Shanks performed very well, whereas the performance of the fourth order Runge-Kutta formula was poor in this case. These results highlight the need for us to be able to derive a variable step-variable order (VSVO) Runge-Kutta algorithm if we are to be able to maintain efficiency over a wide range of tolerances. Our experience on stiff and nonstiff problems indicates that, while in the stiff case it is vital to be able to change order, in the nonstiff case it is often more important to be able to select the correct order initially and to keep this order fixed. The algorithm which we will describe attempts to choose the 'optimal' order initially and then monitors the possibility of changing this order as the integration proceeds. The VSVO algorithms which we derive in this paper have been implemented on some test problems, and the results obtained indicate the superiority of the VSVO approach over conventional fixed order Runge-Kutta formulae.

In the first part of this paper we will derive some explicit formulae for the numerical integration of nonstiff problems and in the second part we will derive diagonally implicit Runge-Kutta (DIRK) formulae suitable for the integration of stiff systems.

2. Some Particular Formulae. All of the integration formulae which we consider in this paper have associated with them a Butcher matrix of the form

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1q} \\ c_2 & a_{21} & \cdots & a_{2q} \\ \vdots & \vdots & & \vdots \\ c_q & a_{q1} & \cdots & a_{qq} \\ \hline & b_1 & \cdots & b_q \end{array} \tag{2.1}$$

We have to modify this notation slightly to describe fully our block formulae, and we will explain these modifications, when they are needed, later in this section. We

first derive a formula of order 2. Butcher has shown that (2.1) has order 2 if

$$\sum_{i=1}^{q} b_i = 1, \qquad \sum_{i=1}^{q} b_i c_i = \frac{1}{2}, \tag{2.2a}$$

and furthermore the principal term in the local truncation error of this formula is

$$\text{PLTE} = \frac{h^3}{6}\left[\left(1 - 3\sum_i b_i c_i^2\right)\{f^2\} + \left(1 - 6\sum_{ij} b_i a_{ij} c_j\right)\{{}_2f\}_2\right], \tag{2.2b}$$

where the terms involving f are the usual elementary differentials. Our formula integrates forward over a step $2h$ and so, in order to be able to use Butcher's analysis, we need to 'normalize' this formula to a step h by dividing all elements of its Butcher matrix by a factor of 2. A normalized 3-stage formula which generalizes the corresponding formula given in [1] is

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1/2 & 1/2 & 0 & \\ \theta/2 & \alpha/2 & \theta/2 - \alpha/2 & 0 \\ \hline & b_1/2 & b_2/2 & 1 - b_1/2 - b_2/2 \end{array} \tag{2.3}$$

Two strategies for choosing the free parameters could be

(1) To improve the computational efficiency of our formulae and/or

(2) To improve the stability properties.

We will give some consideration to aim (2) but mainly in this paper we will be concerned with (1).

We start off our investigation by considering a more or less obvious approach to the problem of satisfying requirement (1). As we shall see, however, this approach yields a rather inefficient formula, and this helps to highlight a problem associated with many existing block formulae. A second order formula requiring only two function evaluations per block can be obtained by taking $\theta = \alpha = 0$ and is given by

$$\begin{aligned} k_1 &= f(x_n, y_n), \qquad k_2 = f(x_{n+1}, y_n + hk_1), \\ y_{n+1}^{(1)} &= y_n + hk_1, \qquad y_{n+2}^{(1)} = y_n + h(k_1 + k_2), \\ y_{n+1}^{(2)} &= y_n + \frac{h}{2}(k_1 + k_2), \qquad y_{n+2}^{(2)} = y_n + 2hk_2. \end{aligned} \tag{2.4}$$

This formula produces second order approximations at both x_{n+1} and x_{n+2} and an estimate of the local truncation error in $y_{n+j}^{(1)}$ is $y_{n+j}^{(2)} - y_{n+j}^{(1)}$ for $j = 1, 2$. If, however, we examine the principal local truncation errors associated with the second order solutions $y_{n+1}^{(2)}$ and $y_{n+2}^{(2)}$, we find that for $y_{n+1}^{(2)}$

$$\text{PLTE} = \frac{h^3}{6}\left[-\tfrac{1}{2}\{f^2\} + \{{}_2f\}_2\right], \tag{2.5a}$$

whereas for $y_{n+2}^{(2)}$

$$\text{PLTE} = \frac{8h^3}{6}\left[\tfrac{1}{4}\{f^2\} + \{{}_2f\}_2\right]. \tag{2.5b}$$

Of course the magnitude of the PLTE in any given situation depends very much on the function being integrated since, in some cases, there may be cancellation amongst the terms appearing in (2.5a) or (2.5b). However, since the constants multiplying the elementary differentials in (2.5b) are larger than those appearing in (2.5a), we would expect (2.5a) to be the more accurate formula in general. (For simple problems where $f(x, y) = ax + by + c$, with a, b and c constants, the PLTE associated with (2.5b) is eight times that associated with (2.5a).) This expectation has been confirmed by practical experience. Furthermore the accuracy obtained with $y_{n+2}^{(2)}$ is exactly that which could have been obtained using a conventional second order Runge-Kutta formula with step $2h$, and so there is not a great deal of advantage in using this block formula. *Thus, when dealing with block formulae, it is vitally important to examine the magnitude of the terms appearing in the PLTE as well as the order of the formula. Thus it is not valid to compare two block formulae of the same order merely by counting the number of stages.* Indeed, we have programmed (2.4) and found it to be generally less efficient than (2.9) even though the latter formula requires one more function evaluation per step.

To overcome this problem we consider an alternative approach which demands an "equi-distribution" of errors in the block scheme. In order to mirror the behavior of conventional single step formulae we construct our block scheme so that if the PLTE in $y_{n+1}^{(2)}$ is of the form $h^3\phi + O(h^4)$, then the PLTE in $y_{n+2}^{(2)}$ is $2h^3\phi + O(h^4)$.

The main reasons for adopting this particular approach are as follows:

(1) By adopting the equi-distribution of error (EDE) approach we have a theoretical reason for expecting our formulae to outperform standard Runge-Kutta formulae. The EDE criterion attempts to ensure that for a smooth problem the local error committed in integrating from x_{n+i} to x_{n+i+1} is the same for all $i \in [0, p-1]$, and this is roughly the behavior of standard Runge-Kutta formula used with fixed h. Thus, if our block formulae can obtain $O(h^{p+1})$ approximations at p grid points using less function evaluations than is required by a pth order R-K formula to perform the same task, we would expect our block formulae to be superior. The results of Section 4 indicate that block formulae developed using this criterion are generally more efficient than standard R-K formulae.

(2) By demanding $O(h^{p+1})$ accuracy at internal points we can obtain $O(h^{p+1})$ solutions at "off-step" points by interpolation. This means that we do not have to choose h so that we "hit" output points, and this offers a significant saving when output is required at many points.

(3) At *each point in the block* we have available a very cheap estimate of the local truncation error in the solution of order $p-1$. By basing our step control on estimates of the error computed at a number of points rather than just one point (see Section 3) we can hope to reduce the likelihood of serious underestimation of the error and hence make the approach more reliable. The results presented in Section 4 do indicate that the block schemes are more reliable than the standard Runge-Kutta formulae.

(4) After considerable numerical experimentation, see particularly [1], it was found that the approach we have adopted gave the best numerical results out of the many possibilities tested. An additional advantage is

(5) By using the analysis given in [11] we can compute an estimate of the stepsize to use after a rejected block.

Thus, if we reconsider formula (2.3), we have the following:

To compute $y_{n+1}^{(1)}$:

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \text{PLTE} = \frac{h^2}{2}\{f\}. \tag{2.6a}$$

To compute $y_{n+2}^{(1)}$:

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & 1 & 1 \end{array}, \qquad \text{PLTE} = h^2\{f\}. \tag{2.6b}$$

To compute $y_{n+1}^{(2)}$:

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \qquad \text{PLTE} = \frac{h^3}{6}\left[-\tfrac{1}{2}\{f^2\} + \{{}_2f\}_2\right]. \tag{2.6c}$$

To compute $y_{n+2}^{(2)}$: use formula (2.3), where for order 2 and for equi-distribution of errors we require

$$\begin{aligned} &b_2/4 + \theta/2(1 - b_1/2 - b_2/2) = 1/2, \\ &1 - 3\left[b_2/8 + \theta^2/4(1 - b_1/2 - b_2/2)\right] = -1/8, \\ &1 - 6\left[(1 - b_1/2 - b_2/2)(\theta/4 - \alpha/4)\right] = 1/4. \end{aligned}$$

One solution to these equations is $\theta = 2$, $\alpha = 0$, $b_1 = \frac{1}{2}$, $b_2 = 1$, $b_3 = \frac{1}{2}$ giving the second order formula

To compute $y_{n+2}^{(2)}$:

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1 & 1 & 0 & \\ 2 & 0 & 2 & 0 \\ \hline & \frac{1}{2} & 1 & \frac{1}{2} \end{array}, \qquad \text{PLTE} = \frac{h^3}{6}\left[-\{f^2\} + \{{}_2f\}_2\right]. \tag{2.6d}$$

The finally accepted second order solutions at $n + 1$ and $n + 2$ are $y_{n+1}^{(2)}$, $y_{n+2}^{(2)}$, and the quantities $y_{n+j}^{(2)} - y_{n+j}^{(1)}$, $j = 1, 2$, serve as an estimate of the local truncation error in $y_{n+j}^{(1)}$. Applying this block formula to the scalar test equation $y' = \lambda y$, we obtain

$$y_{n+2}/y_n = 1 + 2q + 2q^2 + q^3, \qquad q = h\lambda,$$

and it can be shown that the interval of absolute stability of this formula is approximately $(-1.5, 0)$. This block formula does have some computational advantages over conventional Runge-Kutta formulae, and we will defer a discussion of these until Section 3.

As an alternative we could adopt strategy (2) and choose the coefficients b_2, α and θ so as to improve the stability properties of our formula. Such formulae are only of limited value—specifically formulae with increased real stability regions can be of

use in the numerical solution of parabolic partial differential equations, while formulae with increased imaginary stability regions are used in the integration of hyperbolic p.d.e.s. The price which we pay is that we lose the equi-distribution of errors property but the gain is a considerable increase in the stability intervals of our formulae. Applying (2.3) to the scalar test equation $y' = \lambda y$ and putting $q = h\lambda$, we obtain

$$\frac{y_{n+2}}{y_n} = 1 + 2q + 2q^2 + 2(\theta - \alpha)\left(1 - \frac{b_1}{2} - \frac{b_2}{2}\right)q^3. \tag{2.7}$$

The problem of choosing the coefficients of q^3 to give an extended *real* interval of absolute stability has been investigated by Riha [18]. From Riha's analysis it follows that if we choose

$$2(\theta - \alpha)\left(1 - \frac{b_1}{2} - \frac{b_2}{2}\right) = 1/2, \tag{2.8}$$

then the interval of absolute stability of our block formula is $(-3.15, 0)$. We now introduce a measure.

$$M_s = \frac{|\text{interval of absolute stability}| \ * \ \text{number of steps in block}}{\text{number of function evaluations}}.$$

This measure is similar to the scaled stability intervals considered by Watts [27] and is a fair way of comparing the stability properties of two different formulae since in effect it measures "the amount of stability" per function evaluation. Thus $M_s = 2.1$ for (2.8) compared with $M_s = 1$ for a conventional second order Runge-Kutta formula. In view of this it is worthwhile to introduce extra function evaluations when integrating problems which are most efficiently integrated using formulae with large intervals of absolute stability.

A final alternative is to choose our free coefficients so that the intercept of the region of absolute stability with the *imaginary* axis is maximized. An investigation of this problem has been carried out by van der Houwen [26] but the analysis is not yet as complete as that given by Riha for the real case. Formulae with an extended imaginary stability interval are important in the method of lines solution of hyperbolic partial differential equations. The resulting hyperbolic system is normally very large, its eigenvalue spectrum lies on or close to the imaginary axis and this spectrum occupies a large section of the imaginary axis. Using an elementary argument it can be shown that, in order to maximize the imaginary interval of absolute stability, we need to take

$$(\theta - \alpha)\left(1 - \frac{b_1}{2} - \frac{b_2}{2}\right) = 1,$$

and for this choice the imaginary interval of absolute stability is $(0, 2i)$.

We conclude this section by listing some block integration formulae of order 1–4. We will list formulae with improved stability properties of order 2 only and leave the

derivation of higher order formulae as an area for future research. In what follows the weights for solutions at internal points in the block are given under the dotted lines. For example the third order formula is

$$y_{n+1} - y_n = h\left\{\frac{1}{4}k_1 + \frac{3}{4}k_3\right\},$$
$$y_{n+2} - y_n = h\left\{\frac{9}{32}k_1 + \frac{21}{32}k_3 + \frac{7}{32}k_4 + \frac{27}{32}k_5\right\},$$
$$y_{n+3} - y_n = h\left\{\frac{105}{504}k_1 + \frac{117}{112}k_3 + \frac{69}{48}k_4 + \frac{39}{126}k_6\right\}.$$

All formulae are obtained using the equi-distribution of error condition and the formulae are given using the standard R-K formalism for block methods with the coefficients of the Butcher arrays for a pth order formula given for the stepsize $H \equiv ph$. We note that the 'M_s factor' is better than conventional Runge-Kutta formulae for orders 2 and 3 but is worse for order 4.

First Order Formula:

$$y_{n+1} - y_n = hf_n.$$

Second Order Formulae:

Equi-distribution of error

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1/2 & 1/2 & 0 & \\ \hline & 1/4 & 1/4 & \\ 1 & 0 & 1 & 0 \\ \hline & 1/4 & 1/2 & 1/4 \end{array} \qquad M_s = 1.03$$

Maximum Real Stability

(2.9)

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1/2 & 1/2 & 0 & \\ \hline & 1/4 & 1/4 & \\ 1 & 1/2 & 1/2 & 0 \\ \hline & 1/2 & 0 & 1/2 \end{array} \qquad M_s = 2.1$$

Maximum Imaginary Stability

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1/2 & 1/2 & 0 & \\ \hline & 1/4 & 1/4 & \\ 1 & 0 & 1 & 0 \\ \hline & 1/2 & 0 & 1/2 \end{array} \qquad \begin{array}{l}\text{Imaginary Stability} \\ \text{–Interval } (0, 2i)\end{array}$$

Third Order Formula:

(2.10)

0	0						
1/3	1/3	0					
2/9	4/27	2/27	0				
	1/12	0	1/4				
2/3	0	2/3	0	0			
4/9	8/81	−8/81	10/27	2/27	0		
	9/96	0	21/96	7/96	27/96		
1	a_1	a_2	a_3	a_4	a_5	0	
	35/504	0	39/112	23/48	0	13/126	

$M_s = 1.05$

$a_1 = 0.9173076923$, $a_2 = -2.807692308$, $a_3 = 0.3923076923$,
$a_4 = 1.073076923$, $a_5 = 1.425$.

The second order formula used to give an error estimate at $n + 3$ is identical with the above formula for the first 5 stages, and the 6th stage is given by

1	a_1	a_2	a_3	a_4	a_5	0
	43/120	1/2	−41/80	109/240	1/10	1/10

Fourth Order Formula:

(2.11)

0	0										
1/4	1/4	0									
1/6	1/9	1/18	0								
1/8	3/32	1/32	0	0							
1/4	−1/8	−1/8	0	1/2	0						
	1/24	0	0	1/6	1/24						
1/3	a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	0					
1/2	a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	0				
1/2	0	1/2	0	0	0	0	0	0			
1/3	2/27	−2/27	5/18	0	0	0	0	1/18	0		
	7/192	0	0	1/5	−1/24	81/320	5/96	0	0		
3/4	$a_{10,1}$	$a_{10,2}$	$a_{10,3}$	0	0	0	0	$a_{10,4}$	$a_{10,5}$	0	
3/4	$a_{11,1}$	$a_{11,2}$	$a_{11,3}$	$a_{11,4}$	$a_{11,5}$	0	0	$a_{11,6}$	$a_{11,7}$	0	0
	b_1	0	b_3	b_4	b_5	0	0	b_6	b_7	b_8	b_9
1	$a_{12,1}$	$a_{12,2}$	$a_{12,3}$	$a_{12,4}$	$a_{12,5}$	0	0	$a_{12,6}$	$a_{12,7}$	0	$a_{12,8}$
	$\bar{b}_1$	0	$\bar{b}_3$	$\bar{b}_4$	$\bar{b}_5$	0	0	$\bar{b}_6$	$\bar{b}_7$	0	$\bar{b}_8$	$\bar{b}_9$

$M_s = 1/3$

with

$a_{61} = -0.006790123455$, $a_{62} = -1/18$, $a_{63} = 1/20$, $a_{64} = 0.2024691358$,
$a_{65} = 0.14320987655$, $a_{71} = -0.077$, $a_{72} = 0.05$, $a_{73} = -0.243$, $a_{74} = 0.856$,
$a_{75} = -0.896$, $a_{76} = 0.81$, $a_{10,1} = 0.6879807692$, $a_{10,2} = -2.105769231$,
$a_{10,3} = 0.2942307692$, $a_{10,4} = 0.804807692$, $a_{10,5} = 1.06875$,

$$a_{11,1} = 1.013599649, \qquad a_{11,2} = -2.373563736,$$
$$a_{11,3} = 2.101305181, \qquad a_{11,4} = -1.808316282$$
$$a_{11,5} = 0.03836597482, \qquad a_{11,6} = 0.8880102464,$$
$$a_{11,7} = 0.8905989659, \qquad b_1 = -0.1013997396,$$
$$b_3 = -1.316476004, \qquad b_4 = 1.4416666667,$$
$$b_5 = -0.0390625000, \qquad b_6 = 0.2234700521,$$
$$b_7 = 0.4508789062, \qquad b_8 = 1.8, \quad b_9 = -1.527232143,$$
$$a_{12,1} = -2.42247142, \qquad a_{12,2} = 2.9828410875,$$
$$a_{12,3} = -1.1939428, \qquad a_{12,4} = 4.073007497,$$
$$a_{12,5} = 1.111468958, \qquad a_{12,6} = 0.5712931975,$$
$$a_{12,7} = -4.733490737, \qquad a_{12,8} = 0.6112942211,$$
$$\bar{b}_1 = 0.0003635169, \qquad \bar{b}_3 = 0.3368658833,$$
$$\bar{b}_4 = 0.0506860373, \qquad \bar{b}_5 = 0.1566864719,$$
$$\bar{b}_6 = 0.493281834, \qquad \bar{b}_7 = -0.3737884895,$$
$$\bar{b}_8 = 0.2384047462, \qquad \bar{b}_9 = 0.0975.$$

The formula used to obtain a third order solution at $n + 4$ is the same as above with the $\bar{b}_i$, $1 \leqslant i \leqslant 9$, replaced by $\hat{b}_i$ where

$$\hat{b}_1 = 0.001663007867, \qquad \hat{b}_3 = 0.3637090175,$$
$$\hat{b}_4 = 0.02844392849, \qquad \hat{b}_5 = 0.1594404918,$$
$$\hat{b}_6 = 0.4957601898, \qquad \hat{b}_7 = -0.3853166355,$$
$$\hat{b}_8 = 0.2391, \qquad \hat{b}_9 = 0.0972.$$

Note that several of these coefficients are rather large. However, paradoxically, the numerous formulae which have been obtained with smaller coefficients have been found to be much less reliable than those given above. We see from this Section that formulae (2.9), (2.10), (2.11) achieve order 2, 3, 4, respectively, but require only $1\frac{1}{2}$, 2, 3 function evaluations per step. This is better than conventional explicit Runge-Kutta formulae of order p which require p function evaluations per step for $p \leqslant 4$.

3. Computational Aspects. All of the formulae given in the previous section are such that a pth order formula integrates forward in a block of p steps. The way in which we have implemented our formulae allows us to change order only after p integration steps have been completed, although it should be possible to modify Gear's approach to enable us to choose a new step if a block is rejected. As we will show in this section, the block approach does have three major computational advantages in that the algorithm used to choose h is likely to be very reliable, our block approach allows us to change order and also to compute solutions at "off-step" points in a simple manner.

The first problem considered is that of error estimation. As is well known, the local truncation error associated with high order Runge-Kutta formulae is generally so complicated that the problem of error estimation has proved to be a difficult one to overcome. Probably the most successful solution of this problem to date has been embedding, which was developed in a series of papers by Fehlberg [7], [8]. This is the approach which we will adopt in his paper. Note that the Fehlberg approach is

applicable since each block formula of order p has embedded formulas of order 1, $2,\ldots,p-1$. In fact, using an interpolation formula, we could obtain embedded error estimates at any point in the subinterval and not only at mesh points.

Denoting the true solution of our differential equation at x_n by $y(x)$ and the numerical solution obtained at x_{n+1} using a pth order Runge-Kutta formula by $y_{n+1,p}$, we have

$$y(x_{n+1}) - y_{n+1,p} = h^{p+1}\phi_p + O(h^{p+2}), \tag{3.1}$$

where ϕ_p is the principal error function associated with the pth order formula and where we have assumed that $y_{n,p}$ is exact. Associated with the pth order formula there is an embedded formula of order $p-1$ which yields a solution $y_{n+1,p-1}$ at x_{n+1} satisfying

$$y(x_{n+1}) - y_{n+1,p-1} = h^{p}\phi_{p-1} + O(h^{p+1}). \tag{3.2}$$

Subtracting these two relations and assuming that the $O(h^{p+2})$, $h^{p+1}\phi_p$ and $O(h^{p+1})$ terms are negligible compared with the other terms, we have

$$h^{p}\phi_{p-1} \simeq y_{n+1,p} - y_{n+1,p-1}. \tag{3.3}$$

Thus the error in the pth order solution has a principal term

$$h^{p+1}\phi_p = h(y_{n+1,p} - y_{n+1,p-1})\frac{\phi_p}{\phi_{p-1}} + O(h^{p+2}).$$

[This assumes that y is a scalar but the argument can be extended to the vector case.] If we control the step length, h, by an estimate of

$$y_{n+1,p} - y_{n+1,p-1} = h^{p+1}\phi_p\left\{\frac{\phi_{p-1}}{h\phi_p}\right\},$$

this corresponds to controlling the error in $y_{n+1,p}$ per unit step; c.f. [24]. The procedure which we now use to adjust the step is as follows. Assuming that a local error tolerance, Tol, is specified and that an error estimate E_i is obtained at $n+i$, $i \in [1, p]$, the step h_1 used to compute forward from $n+p$ is (assuming $\|E_{i+1} - E_i\| < \text{Tol}$, for all $i \in [1, p-1]$),

$$h_1 = \text{SF} * h * \left(\text{Tol}/\|\hat{E}\|\right)^{1/p+1}, \qquad \hat{E} = \max_{0\leqslant i\leqslant p-1} \|E_{i+1} - E_i\|, \tag{3.4}$$

where $\|\cdot\|$ is some convenient vector norm and SF is a safety factor. If $\|\hat{E}\| > \text{Tol}$, the current block is abandoned and the integration is restarted from the point x_n using a step h_1 given by (3.4). In our practical experiments we took $\|\cdot\|$ as the maximum norm, and if $\|\hat{E}\| < \text{Tol}$, we performed local extrapolation. A procedure whereby local extrapolation is always performed may not be the most efficient, especially for low accuracy requirements, but it is a convenient procedure for allowing us to compare our block formulae with more conventional formulae.

The second problem which we consider is that of computing a solution at an "off-step" point. If we denote by G_n the set of points to be used by the integration formula and by S_n the set of points at which output is required, the normal procedure is to adjust h so that $S_n \subset G_n$. However, this procedure can be very wasteful especially if S_n contains many points some of which are closely spaced (see, for example, [23]). This output problem is not experienced by linear multistep

methods since they compute intermediate solutions essentially by evaluating an interpolating polynomial passing through already computed solutions. Interpolation can also be used with block Runge-Kutta formulae to compute intermediate solutions. Having completed a block of p integration steps, we have $p + 1$ solutions $y_n, y_{n+1}, \ldots, y_{n+p}$ available. If we now fit a pth order interpolating polynomial $P_h(x)$ through the points (x_i, y_i), $i = n, n + 1, \ldots, n + p$, we can compute a pth order solution y_ν at any intermediate point x_ν as $y_\nu = P_h(x_\nu)$.

Finally, in this section, we consider an algorithm for changing the order of our block implicit Runge-Kutta formulae. Comparison of codes by various workers have shown that in some situations Runge-Kutta formulae are very valuable, especially when function evaluations are inexpensive or when a problem has a number of discontinuities which require 'restarts', if some care is taken to choose the correct order initially and if the possibility of changing order is monitored as the integration proceeds. In what follows we describe an algorithm which allows us to achieve this aim. The algorithm which we describe has proved to be reasonably successful in practice, but it cannot hope to be as good as order changing algorithms for linear multistep methods. The reason for this is that successful codes based on linear multistep methods are able to compute an estimate of the error which would have been committed if a formula of degree one higher than the current one had been used, without actually having to compute the higher order solution (cf. [10], [13], [24]). They are able to do this because the form of the local truncation error associated with linear multistep methods is particularly simple. However, this facility does not seem possible with Runge-Kutta formulae, and the problem of when to *increase* order is a stumbling block. The procedure which we use is an extension of one given by Cash and Liem [5] for DIRK formulae. The first thing we need to do is to work out the order of the formula we expect to use. As a result of extensive numerical experiments we have found that in the range 10^{-1} to 10^{-9} we expect to use the following order formulae:

$$10^{-1}, 2; \quad 10^{-2}, 2; \quad 10^{-3}, 3; \quad 10^{-4}, 3; \quad 10^{-5} - 10^{-9}, 4.$$

Now, initially, we can choose the optimal formula to use by computing a fourth order solution with step h together with the embedded 3rd, 2nd and 1st order solutions. From this we can compute

$$E_i = \text{estimated error per unit step in the } i\text{th order solution } i = 2, 3, 4$$

and

$$h_i = h \times \mathrm{SF} \times \left((\mathrm{Tol}/E_i)\right)^{1/i} \text{ as the next step which could be attempted.}$$

Clearly if SF is taken too large it will result in inefficiency because of many rejected steps, and if SF is too small, inefficiency will occur as a result of the step being too far from optimum. Numerical experiment showed that a suitable choice for SF was $\mathrm{SF} = 0.8$ for order 2, 3 and $\mathrm{SF} = 0.9$ for order 4. Having calculated h_i, we can compute

$$w_2 = 2 \times h_2/3, \quad w_3 = 3 \times h_3/6 \quad \text{and} \quad w_4 = 4 \times h_4/12.$$

The quantity w_i measures the distance that can be integrated forward per function evaluation, and the order chosen is j where

$$w_j = \max_i w_i.$$

We now have to deal with two possibilities:

(a) If j is equal to or greater than the expected order, we integrate forward one block and then examine the possibility of reducing the order by computing the embedded solutions. We do not attempt to increase order.

(b) If j is less than the expected order, we integrate forward one block and examine the possibility of reducing order. If our test tells us not to reduce order, we increase the order by 1 and then integrate another block forward.

This process is carried out at the end of each block. Although this procedure is not ideal it has proved to be quite successful on the test problems which we have considered as the results of the next section show.

4. Numerical Results. In this section we present some numerical results obtained using the block Runge-Kutta formulae developed in Section 2. There are three main points which we wish to make. First, we wish to show that block Runge-Kutta formulae are reliable for the integration of nonstiff problems. Secondly, we wish to show that fixed order block formulae are competitive with conventional Runge-Kutta formulae of the same order, and lastly we wish to show that our variable order algorithm is both efficient and reliable. The test problems considered are sets A, B, D and E given by Hull et al. [14], and these consist of twenty test problems. These problems were run for tolerances 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, and the results obtained are given in Tables 1–7. The table headings are self-explanatory except, perhaps, percent deceived which counts the number of *accepted* solutions with errors greater than the specified tolerance, and maximum error which gives the maximum absolute error as a fraction of the tolerance in an accepted solution. It can be seen from Tables 1–6 that the fixed order block schemes are generally more efficient than the fixed order Runge-Kutta formulae and are more reliable. Also the variable order formula performs well and is more reliable over all tolerances than any fixed order scheme. The step control for conventional R-K was performed as described in [14, p. 622].

We finish with a few remarks regarding the implementation of our block formulae. First, we note that we have allowed the Runge-Kutta formulae to change step size at the end of each integration step, whereas block schemes can only change step size at the end of a block, unless the block is aborted. For the very inexpensive functions of the test set, step size changing may represent a considerable overhead, but this overhead is not listed in the Tables of results. Secondly, we have always insisted that a block formula completes one block step forward before computing error estimates. For the fourth order formula, for example, a bad step results in 12 wasted function evaluations. It is possible to abort much earlier if things are going wrong, but we have not investigated this possibility. Finally, we note from the results that often we achieve more accuracy than is required especially with the fourth order formula. This indicates that more research into efficient control of step length for block formulae would be valuable.

Acknowledgement. The author is grateful to the referee for many comments and suggestions.

TABLE 1

Results for second order block formula for tolerances 10^{-1}, 10^{-3}

	FCN CALLS	No. of Steps	Steps deceived	Max. Error
Class A	1743	1112	4	1.15
Class B	4245	2730	1	1.01
Class D	6981	4556	5	1.19
Class E	4566	2948	1	1.07

TABLE 2

Results for second order Runge-Kutta for tolerances 10^{-1}, 10^{-3}

Class A	2228	1087	5	1.26
Class B	5402	2668	6	1.12
Class D	9306	4463	17	1.19
Class E	5880	2909	1	1.10

TABLE 3

Results for third order Runge-Kutta for tolerances 10^{-1}, 10^{-3}, 10^{-5}

Class A	3933	1239	2	1.55
Class B	10629	3276	2	1.08
Class D	18552	5511	0	0.96
Class E	10512	3664	2	1.17

TABLE 4

Results for third order block formula for tolerances 10^{-1}, 10^{-3}, 10^{-5}

Class A	3180	1551	0	0.94
Class B	7356	3522	0	0.91
Class D	14834	7170	0	0.91
Class E	7980	3822	0	0.86

TABLE 5

Results for fourth order block scheme

	CLASS A					CLASS B					CLASS D					CLASS E			
10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error
A1	72	20	0	.189	B1	420	96	0	.987	D1	144	36	0	.134	E1	96	32	0	.219
A2	36	12	0	.367	B2	228	64	0	.079	D2	192	44	0	.114	E2	696	136	.05	1.616
A3	120	32	0.06	1.600	B3	108	28	0	.114	D3	216	48	0	.087	E3	264	72	.04	2.063
A4	48	12	0	.249	B4	1260	304	0	.421	D4	84	28	0	.254	E4	36	12	0	0.912
A5	48	16	0	.039	B5	252	56	0	.194	D5	108	36	0	.216	E5	24	8	0	0.830
10^{-3}					10^{-3}					10^{-3}					10^{-3}				
A1	96	32	0	.125	B1	804	184	0	.372	D1	432	116	0	.384	E1	204	68	0	.249
A2	72	24	0	.113	B2	264	76	0	.196	D2	600	144	0	.402	E2	828	200	0	.418
A3	384	96	0	.507	B3	132	40	0	.076	D3	768	180	0	.297	E3	504	148	0	.363
A4	96	24	0	.223	B4	456	120	0	.296	D4	1044	240	0	.248	E4	48	16	0	.279
A5	72	24	0	.190	B5	360	92	0	.266	D5	1536	356	0	.222	E5	84	20	0	.460
10^{-5}					10^{-5}					10^{-5}					10^{-5}				
A1	192	64	0	.386	B1	1632	424	0	.447	D1	1044	348	0	.309	E1	552	180	0	.291
A2	144	48	0	.094	B2	336	104	0	.436	D2	1140	356	0	.347	E2	1812	476	0	.519
A3	900	248	0	.551	B3	240	76	0	.267	D3	1536	404	0	.336	E3	1392	432	0	.423
A4	192	60	0	.353	B4	960	304	0	.333	D4	1920	488	0	.343	E4	108	36	0	.421
A5	180	60	0	.246	B5	780	220	0	.287	D5	2820	696	0	.373	E5	168	144	0	.773
10^{-7}					10^{-7}					10^{-7}					10^{-7}				
A1	468	156	0	.460	B1	3660	1128	0	.409	D1	3276	1092	0	.246	E1	1584	528	0	.256
A2	336	112	0	.163	B2	612	204	0	.464	D2	3312	1104	0	.269	E2	4128	1332	0	.552
A3	2292	728	0	.634	B3	552	184	0	.596	D3	3360	1120	0	.315	E3	4044	1348	0	.307
A4	540	176	0	.279	B4	2664	888	0	.463	D4	4140	1252	0	.696	E4	312	104	0	.326
A5	516	172	0	.236	B5	1980	616	0	.468	D5	*				E5	432	144	0	.353
10^{-9}					10^{-9}					10^{-9}					10^{-9}				
A1	1404	468	0	.468	B1	*				D1	*				E1	4920	1640	0	.236
A2	948	316	0	.195	B2	1836	612	0	.467	D2	*				E2	*			
A3	6804	2268	0	.531	B3	1704	568	0	.346	D3	*				E3	*			
A4	1632	544	0	.381	B4	*				D4	*				E4	996	332	0	.251
A5	1572	524	0	.239	B5	*				D5	*				E5	1440	480	0	.235

TABLE 6

Results for fourth order Runge-Kutta formula

	CLASS A					CLASS B					CLASS D					CLASS E			
10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error
A1	55	10	0	0.31	B1	245	37	0.06	1.45	D1	165	33	0	0.86	E1	75	15	0	0.44
A2	30	6	0	0.08	B2	125	24	0	0.45	D2	225	36	0	0.58	E2	270	42	0.1	1.08
A3	135	20	0	0.81	B3	60	11	0	0.27	D3	265	41	0	0.77	E3	180	29	0.1	1.06
A4	30	6	0	0.67	B4	155	27	0	0.89	D4	310	47	0	0.75	E4	30	4	0	0.31
A5	25	5	0	0.34	B5	120	19	0	0.73	D5	300	47	0.05	1.22	E5	30	4	0	0.81
10^{-3}					10^{-3}					10^{-3}					10^{-3}				
A1	90	18	0	0.39	B1	545	104	0.05	1.08	D1	450	90	0	0.57	E1	225	455	0	0.55
A2	55	11	0	0.39	B2	170	33	0	0.38	D2	465	93	0	0.84	E2	695	119	0.01	1.01
A3	370	61	0.02	1.06	B3	105	21	0	0.52	D3	535	101	0.01	1.01	E3	465	87	0	0.91
A4	90	17	0	0.59	B4	395	78	0	0.82	D4	715	119	0	0.98	E4	60	10	0	0.65
A5	55	11	0	0.52	B5	275	55	0	0.79	D5	990	159	0	0.90	E5	75	10	0.1	1.07
10^{-5}					10^{-5}					10^{-5}					10^{-5}				
A1	230	46	0	0.40	B1	1645	325	0.01	1.09	D1	1400	280	0	0.47	E1	710	142	0	0.54
A2	135	27	0	0.35	B2	325	63	0	0.40	D2	1455	291	0	0.51	E2	1795	359	0.01	1.06
A3	990	183	0	0.92	B3	275	55	0	0.45	D3	1560	312	0	0.59	E3	1355	271	0	0.88
A4	250	49	0	0.80	B4	1270	251	0	0.74	D4	1755	351	0	0.69	E4	135	25	0	0.81
A5	165	33	0	0.82	B5	870	174	0	0.50	D5	2245	449	0	0.88	E5	155	30	0.04	1.28
10^{-7}					10^{-7}					10^{-7}					10^{-7}				
A1	665	133	0	0.41	B1	5105	1021	0	0.87	D1	4420	884	0	0.47	E1	2230	446	0	0.58
A2	375	75	0	0.35	B2	870	174	0	0.40	D2	4590	918	0	0.44	E2	5665	1133	0	0.67
A3	2910	567	0	0.98	B3	810	162	0	0.46	D3	4905	981	0	0.46	E3	4290	858	0	0.61
A4	770	153	0	0.80	B4	3970	791	0	0.75	D4	5520	1104	0	0.48	E4	380	75	0	0.93
A5	520	103	0	0.76	B5	2745	549	0	0.50	D5	7070	1414	0	0.57	E5	475	95	0	0.83
10^{-9}					10^{-9}					10^{-9}					10^{-9}				
A1	2030	406	0	0.41	B1	*				D1	*				E1	7035	1407	0	0.59
A2	1130	226	0	0.39	B2	2655	531	0	0.41	D2	*				E2	*			
A3	8990	1782	0	0.96	B3	2505	501	0	0.46	D3	*				E3	13560	2712	0	0.47
A4	2425	484	0	0.82	B4	*				D4	*				E4	1180	235	0	0.67
A5	1615	322	0.01	1.07	B5	*				D5	*				E5	1525	305	0	0.55

TABLE 7

Results for variable order block scheme

	CLASS A					CLASS B					CLASS D					CLASS E			
10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-1}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error	10^{-}	FCN Calls	No. of Steps	Percent Deceived	Maximum Error
A1	21	14	0	.570	B1	216	112	0.02	1.296	D1	75	49	0	.979	E1	48	32	0	.608
A2	24	12	0	.184	B2	87	42	0	.154	D2	87	54	.04	1.136	E2	288	137	0	.767
A3	117	58	.05	1.218	B3	36	21	0	.244	D3	126	68	.03	1.224	E3	126	63	0	.529
A4	36	10	0	.259	B4	135	71	0	.931	D4	72	24	0	.696	E4	30	9	0	.159
A5	36	12	0	.084	B5	126	45	0	.517	D5	102	33	0	.552	E5	36	8	0	.218
10^{-3}					10^{-3}					10^{-3}					10^{-3}				
A1	81	31	0	.126	B1	642	264	0	.784	D1	414	207	0	.732	E1	204	68	0	.249
A2	60	30	0	.286	B2	171	60	0	.236	D2	600	144	0	.402	E2	708	291	0	.829
A3	342	141	0	.873	B3	96	48	0	.391	D3	780	180	0	.297	E3	504	252	0	.663
A4	84	24	0	.360	B4	456	120	0	.316	D4	1056	240	0	.248	E4	48	21	0	.397
A5	72	24	0	.156	B5	342	132	0	.812	D5	1560	356	0	.223	E5	72	16	0	.335
10^{-5}					10^{-5}					10^{-5}					10^{-5}				
A1	192	64	0	.386	B1	1614	423	0	.418	D1	1044	348	0	.309	E1	552	180	0	.291
A2	144	48	0	.093	B2	336	104	0	.436	D2	1140	356	0	.347	E2	1818	479	0	.516
A3	900	248	0	.551	B3	240	76	0	.267	D3	1548	404	0	.336	E3	1410	435	0	.409
A4	192	60	0	.353	B4	960	304	0	.333	D4	1932	488	0	.343	E4	108	36	0	.420
A5	180	60	0	.246	B5	780	220	0	.287	D5	2832	696	0	.374	E5	144	40	0	.852
10^{-7}					10^{-7}					10^{-7}					10^{-7}				
A1	468	156	0	.460	B1	3660	1128	0	.409	D1	3276	1092	0	.246	E1	1584	528	0	.256
A2	336	112	0	.163	B2	612	204	0	.464	D2	3312	1104	0	.269	E2	4134	1335	0	.563
A3	2292	728	0	.634	B3	552	184	0	.596	D3	3360	1120	0	.315	E3	4044	1348	0	.307
A4	540	176	0	.279	B4	2664	888	0	.463	D4	4152	1252	0	.696	E4	312	104	0	.326
A5	516	172	0	.236	B5	1980	616	0	.468	D5	*				E5	432	144	0	.353
10^{-9}					10^{-9}					10^{-9}					10^{-9}				
A1	1404	468	0	.468	B1	*				D1	*				E1	4920	1640	0	.236
A2	948	316	0	.195	B2	1836	612	0	.467	D2	*				E2	*			
A3	6804	2268	0	.531	B3	1704	568	0	.346	D3	*				E3	*			
A4	1632	544	0	.381	B4	*				D4	*				E4	996	332	0	.251
A5	1572	524	0	.239	B5	*				D5	*				E5	1440	480	0	.235

Department of Mathematics
Imperial College
South Kensington
London S.W.7, England

1. J. BOND, *Some Block Iterative Methods for the Numerical Solution of Systems of Ordinary Differential Equations*, Ph. D. thesis, Univ. of London, 1979.

2. D. G. BRUSH, J. J. KOHFELD & G. T. THOMPSON, "Solution of ordinary differential equations using two 'off-step' points," *J. Assoc. Comput. Mach.*, v. 14, 1967, pp. 769–784.

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

4. J. C. BUTCHER, "Coefficients for the study of Runge-Kutta integration processes," *J Austral. Math. Soc.*, v. 3, 1963, pp. 185–201.

5. J. R. CASH & C. B. LIEM, "On the design of a variable order variable step diagonally implicit Runge-Kutta algorithm," *J. Inst. Math. Appl.*, v. 26, 1980, pp. 87–91.

6. J. DONELSON & E. HANSEN, "Cyclic composite multistep predictor-corrector methods," *SIAM J. Numer. Anal.*, v. 8, 1971, pp. 37–157.

7. E. FEHLBERG, *Classical Fifth, Sixth, Seventh and Eighth Order Runge-Kutta Formulas With Stepsize Control*, NASA technical report no. 287, 1968.

8. E. FEHLBERG, *Low Order Classical Runge-Kutta Formulas With Stepsize Control and Their Application to Some Heat Transfer Problems*, NASA technical report no. 315, 1969.

9. L. FOX, *The Numerical Solution of Two-Point Boundary Value Problems in O.D.E.s*, Oxford Univ. Press, New York, 1957.

10. C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N. J., 1971.

11. C. W. GEAR, "Runge-Kutta starters for multistep methods," *ACM Trans. Math. Software*, v. 6, 1980, pp. 263–279.

12. W. B. GRAGG & H. J. STETTER, "Generalised multistep predictor-corrector methods," *J. Assoc. Comput. Mach.*, v. 11, 1964, pp. 188–209.

13. A. C. HINDMARSH, *GEAR: Ordinary Differential Equation System Solver*, UCID-30001, Rev. 3, Lawrence Livermore Laboratory, Univ. of California, 1974.

14. T. E. HULL, W. H. ENRIGHT, B. M. FELLEN & A. E. SEDGWICK, "Comparing numerical methods for ordinary differential equations," *SIAM J. Numer. Anal.*, v. 9, 1972, pp. 603–637.

15. T. E. HULL & W. H. ENRIGHT, "Test results on initial value methods for non-stiff O.D.E.s," *SIAM J. Numer. Anal.*, v. 13, 1976, pp. 944–961.

16. J. J. KOHFELD & G. T. THOMPSON, "Multistep methods with modified predictors," *J. Assoc. Comput. Mach.*, v. 14, 1967, pp. 155–166.

17. A. ISERLES, "On the A-stability of implicit Runge-Kutta processes," *BIT*, v. 18, 1978, pp. 157–169.

18. W. RIHA, "Optimal stability polynomials," *Computing*, v. 9, 1972, pp. 37–43.

19. J. BARKLEY ROSSER, "A Runge-Kutta for all seasons," *SIAM Rev.*, v. 9, 1967, pp. 417–452.

20. D. SARAFYAN, *Composite and Multi-Step Runge-Kutta Formulas*, Technical Report No. 18, Louisiana State University, Nov. 1966.

21. D. SARAFYAN, *Multi-Order Property of Runge-Kutta Formulas and Error Estimation*, Technical Report No. 29, Louisiana State University, Nov. 1967.

22. L. F. SHAMPINE & H. A. WATTS, "Global error estimation for ordinary differential equations," *ACM Trans. Math. Software*, v. 2, 1976, pp. 172–186.

23. L. F. SHAMPINE, M. K. GORDON & J. A. WISNIEWSKI, "Variable Order Runge-Kutta codes," *Computational Techniques for Ordinary Differential Equations* (I. Gladwell and D. K. Sayers, eds.) Academic Press, London, 1980, pp. 83–101.

24. L. F. SHAMPINE & M. K. GORDON, *Computer Solution of Ordinary Differential Equations*, Freeman, San Francisco, 1975.

25. H. J. STETTER, *Analysis of Discretization Methods for Ordinary Differential Equations*, Springer-Verlag, Berlin and New York, 1973.

26. P. J. VAN DER HOUWEN, *Construction of Integration Formulas for Initial Value Problems*, North-Holland, Amsterdam, 1976.

27. H. A. WATTS, *Runge-Kutta-Fehlberg Methods: Scaled Stability Regions*, report number SAND76-0323, 1976.

28. J. WILLIAMS & F. DE HOOG, "A class of A-stable advanced multistep methods," *Math. Comp.*, v. 28, 1974, pp. 163–177.