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

\[
\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0, y \in \mathbb{R}^r.
\]

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 \( p \)th order explicit Runge-Kutta methods integrating forward over a single step \( H = ph \) 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...
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 (\( \leq 4 \)) 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 (VSSVO) 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 VSSVO algorithms which we derive in this paper have been implemented on some test problems, and the results obtained indicate the superiority of the VSSVO 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

\[
\begin{array}{c|cccc}
  c_1 & a_{11} & a_{12} & \cdots & a_{1q} \\
  c_2 & a_{21} & a_{22} & \cdots & a_{2q} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  c_q & a_{q1} & a_{q2} & \cdots & a_{qq} \\
  \hline
  b_1 & b_2 & \cdots & b_q
\end{array}
\]

(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

\[ \sum_{i=1}^{q} b_i = 1, \quad \sum_{i=1}^{q} b_i c_i = \frac{1}{2}, \]

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

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

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

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

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

\[
\begin{align*}
&k_1 = f(x_n, y_n), \\
&k_2 = f(x_{n+1}, y_n + h k_1), \\
y^{(1)}_{n+1} = y_n + h k_1, & y^{(1)}_{n+2} = y_n + h (k_1 + k_2), \\
y^{(2)}_{n+1} = y_n + \frac{h}{2} (k_1 + k_2), & y^{(2)}_{n+2} = y_n + 2hk_2.
\end{align*}
\]

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

\[ \text{PLTE} = \frac{h^3}{6} \left[ -\frac{1}{2} \{ f^2 \} + \{ 2f \} \right], \]

whereas for \( y^{(2)}_{n+2} \)

\[ \text{PLTE} = \frac{8h^3}{6} \left[ \frac{1}{4} \{ f^2 \} + \{ 2f \} \right]. \]
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 \( y_{i+2}^{(2)} \) 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 \( y_{i+1}^{(2)} \) is of the form \( h^3\phi + O(h^4) \), then the PLTE in \( y_{i+2}^{(2)} \) 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 \( p \)th 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{equation}
0 \begin{bmatrix}
0 \\
1
\end{bmatrix}, \quad \text{PLTE} = \frac{h^2}{2} \{ f \}.
\end{equation}

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

\begin{equation}
0 \begin{bmatrix}
0 \\
1 \\
1
\end{bmatrix}, \quad \text{PLTE} = h^2 \{ f \}.
\end{equation}

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

\begin{equation}
0 \begin{bmatrix}
0 \\
1 \\
1 \\
\frac{1}{2}
\end{bmatrix}, \quad \text{PLTE} = \frac{h^3}{6} \left[ -\frac{1}{2} \{ f^2 \} + \{ z_2 f \} \right].
\end{equation}

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

\begin{align*}
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 - a/4) \right] &= 1/4.
\end{align*}

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

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

\begin{equation}
0 \begin{bmatrix}
0 \\
1 \\
2 \\
\frac{1}{2}
\end{bmatrix}, \quad \text{PLTE} = \frac{h^3}{6} \left[ - \{ f^2 \} + \{ z_2 f \} \right].
\end{equation}

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

\[ y_{n+2}/y_n = 1 + 2q + 2q^2 + q^3, \quad q = h\lambda, \]

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 $b_2$, $a$ and $\theta$ 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

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

The problem of choosing the coefficients of \( q^3 \) to give an extended \textit{real} interval of absolute stability has been investigated by Riha [18]. From Riha’s analysis it follows that if we choose

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

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

\[
M_s = \frac{\text{interval of absolute stability}}{\text{number of steps in block}} \times \text{number of function evaluations}.
\]

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 \textit{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 \( p \) th order formula given for the stepsize \( H = ph \). We note that the 'Ms 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**

\[
\begin{array}{c|ccc}
0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
1/4 & 1/4 & 1 & 0 \\
\hline
1/4 & 1/2 & 1/4
\end{array}
\]

\( M_s = 1.03 \)

**Maximum Real Stability**

\[
\begin{array}{c|ccc}
0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
1/4 & 1/4 & 1/4 \\
1 & 1/2 & 1/2 & 0 \\
\hline
1/2 & 0 & 1/2
\end{array}
\]

\( M_s = 2.1 \)

**Maximum Imaginary Stability**

\[
\begin{array}{c|ccc}
0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
1/4 & 1/4 & 1/4 \\
1 & 0 & 1 & 0 \\
\hline
1/2 & 0 & 1/2
\end{array}
\]

Imaginary Stability — Interval \( (0, 2i) \)
**Third Order Formula:**

\[
\begin{array}{c|c|c|c|c|c|c}
0 & 0 & 0 \\
1/3 & 1/3 & 0 \\
2/9 & 4/27 & 2/27 & 0 \\
\hline
& & & & & \end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
1/12 & 0 & 1/4 \\
2/3 & 0 & 2/3 & 0 & 0 \\
4/9 & 8/81 & -8/81 & 10/27 & 2/27 & 0 \\
\hline
1 & a_1 & a_2 & a_3 & a_4 & a_5 & 0 \\
\hline
35 & 0 & 39 & 23 & 13 \\
504 & 0 & 112 & 48 & 0 & 126 \\
\end{array}
\]

\[M_s = 1.05\]

\[a_1 = 0.9173076923, \quad a_2 = -2.807692308, \quad a_3 = 0.3923076923,\]
\[a_4 = 1.073076923, \quad a_5 = 1.425.\]

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

\[
\begin{array}{c|c|c|c|c|c|c}
1 & a_1 & a_2 & a_3 & a_4 & a_5 & 0 \\
\hline
43/120 & 1/2 & -41/80 & 109/240 & 1/10 & 1/10 \\
\end{array}
\]

**Fourth Order Formula:**

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c}
0 & 0 & 0 \\
1/4 & 1/4 & 0 \\
1/6 & 1/9 & 1/18 & 0 \\
1/8 & 3/32 & 1/32 & 0 & 0 \\
1/4 & -1/8 & -1/8 & 0 & 1/2 & 0 \\
\hline
1/24 & 0 & 0 & 1/6 & 1/24 \\
1/3 & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & 0 \\
1/2 & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & 0 \\
1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 2/27 & -2/27 & 5/18 & 0 & 0 & 0 & 0 & 1/18 & 0 \\
\hline
3/4 & a_{10,1} & a_{10,2} & a_{10,3} & 0 & 0 & 0 & 0 & a_{10,4} & a_{10,5} & 0 \\
3/4 & a_{11,1} & a_{11,2} & a_{11,3} & a_{11,4} & a_{11,5} & 0 & 0 & a_{11,6} & a_{11,7} & 0 & 0 \\
\hline
1 & a_{12,1} & a_{12,2} & a_{12,3} & a_{12,4} & a_{12,5} & 0 & 0 & b_6 & b_7 & b_8 & b_9 \\
\hline
& b_1 & b_2 & b_3 & b_4 & b_5 & 0 & 0 & b_6 & b_7 & b_8 & b_9 \\
\end{array}
\]

\[M_s = 1/3\]

\[a_{61} = -0.006790123455, \quad a_{62} = -1/18, \quad a_{63} = 1/20, \quad a_{64} = 0.2024691358,\]
\[a_{65} = 0.14320987655, \quad a_{71} = -0.077, \quad a_{72} = 0.05, \quad a_{73} = -0.243, \quad a_{74} = 0.856,\]
\[a_{75} = -0.896, \quad a_{76} = 0.81, \quad a_{10,1} = 0.6879807692, \quad a_{10,2} = -2.105769231,\]
\[a_{10,3} = 0.2942307692, \quad a_{10,4} = 0.804807692, \quad a_{10,5} = 1.06875,\]
The formula used to obtain a third order solution at \( n + 4 \) is the same as above with the \( \tilde{b}_i \), \( 1 \leq i \leq 9 \), replaced by \( \tilde{b}_i \) where

\[
\begin{align*}
\tilde{b}_1 &= 0.001663007867, \\
\tilde{b}_2 &= 0.02844392849, \\
\tilde{b}_3 &= 0.4957601898, \\
\tilde{b}_4 &= 0.2391,
\end{align*}
\]

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, 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 \( p \leq 4 \).

3. Computational Aspects. All of the formulae given in the previous section are such that a \( p \)th 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 \( p \)th order Runge-Kutta formula by \( y_{n+1,p} \), we have

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

where \( \phi_p \) is the principal error function associated with the \( p \)th order formula and where we have assumed that \( y_{n,p} \) is exact. Associated with the \( p \)th 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

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

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

\[
h p \phi_{p-1} \approx y_{n+1,p-1} - y_{n+1,p}.
\]

Thus the error in the \( p \)th order solution has a principal term

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

[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

\[
y_{n+1,p} - y_{n+1,p-1} = h^{p+1} \phi_p \frac{\phi_{p-1}}{h \phi_p},
\]

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 \( h_i \) used to compute forward from \( n + p \) is (assuming \( \| E_{i+1} - E_i \| < Tol \) for all \( i \in [1, p - 1] \),

\[
h_i = SF \cdot h \cdot (\| \hat{E} \|)^{1/p+1}, \quad \hat{E} = \max_{0 \leq i < p-1} \| E_{i+1} - E_i \|,
\]

where \( \| \cdot \| \) is some convenient vector norm and \( SF \) is a safety factor. If \( \| \hat{E} \| \gg Tol \), 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 \( \| \hat{E} \| < Tol \), 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 \( p \)th order interpolating polynomial \( P_p(x) \) through the points \((x_i, y_i), i = n, n+1, \ldots, n+p\), we can compute a \( p \)th order solution \( y_p \) at any intermediate point \( x_p \) as \( y_p = P_p(x_p) \).

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 of 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

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

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 \( SF = 0.8 \) for order 2, 3 and \( SF = 0.9 \) 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 $10^{-1}, 10^{-3}$

<table>
<thead>
<tr>
<th>FCN CALLS</th>
<th>No. of Steps</th>
<th>Steps</th>
<th>Max. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>1743</td>
<td>1112</td>
<td>4</td>
</tr>
<tr>
<td>Class B</td>
<td>4245</td>
<td>2730</td>
<td>1</td>
</tr>
<tr>
<td>Class D</td>
<td>6981</td>
<td>4556</td>
<td>5</td>
</tr>
<tr>
<td>Class E</td>
<td>4566</td>
<td>2948</td>
<td>1</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th></th>
<th>CLASS A</th>
<th></th>
<th>CLASS B</th>
<th></th>
<th>CLASS D</th>
<th></th>
<th>CLASS E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10</strong></td>
<td>FCN</td>
<td>No. of</td>
<td>Percent</td>
<td>Maximum</td>
<td>FCN</td>
<td>No. of</td>
<td>Percent</td>
<td>Maximum</td>
</tr>
<tr>
<td></td>
<td>Calls</td>
<td>Steps</td>
<td>Deceived</td>
<td>Error</td>
<td>Calls</td>
<td>Steps</td>
<td>Deceived</td>
<td>Error</td>
</tr>
<tr>
<td>A1</td>
<td>72</td>
<td>20</td>
<td>0</td>
<td>.189</td>
<td>420</td>
<td>96</td>
<td>0</td>
<td>.987</td>
</tr>
<tr>
<td>A2</td>
<td>36</td>
<td>12</td>
<td>0</td>
<td>.367</td>
<td>228</td>
<td>64</td>
<td>0</td>
<td>.079</td>
</tr>
<tr>
<td>A3</td>
<td>120</td>
<td>32</td>
<td>.06</td>
<td>1.600</td>
<td>108</td>
<td>28</td>
<td>0</td>
<td>.114</td>
</tr>
<tr>
<td>A4</td>
<td>48</td>
<td>12</td>
<td>0</td>
<td>.249</td>
<td>1260</td>
<td>304</td>
<td>0</td>
<td>.421</td>
</tr>
<tr>
<td>A5</td>
<td>48</td>
<td>16</td>
<td>0</td>
<td>.039</td>
<td>252</td>
<td>56</td>
<td>0</td>
<td>.194</td>
</tr>
<tr>
<td>A1</td>
<td>96</td>
<td>32</td>
<td>0</td>
<td>.125</td>
<td>804</td>
<td>184</td>
<td>0</td>
<td>.372</td>
</tr>
<tr>
<td>A2</td>
<td>72</td>
<td>24</td>
<td>0</td>
<td>.113</td>
<td>264</td>
<td>76</td>
<td>0</td>
<td>.196</td>
</tr>
<tr>
<td>A3</td>
<td>384</td>
<td>96</td>
<td>0</td>
<td>.507</td>
<td>132</td>
<td>40</td>
<td>0</td>
<td>.076</td>
</tr>
<tr>
<td>A4</td>
<td>96</td>
<td>24</td>
<td>0</td>
<td>.223</td>
<td>456</td>
<td>120</td>
<td>0</td>
<td>.296</td>
</tr>
<tr>
<td>A5</td>
<td>72</td>
<td>24</td>
<td>0</td>
<td>.190</td>
<td>360</td>
<td>92</td>
<td>0</td>
<td>.266</td>
</tr>
</tbody>
</table>

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use
<table>
<thead>
<tr>
<th>CLASS A</th>
<th>CLASS B</th>
<th>CLASS D</th>
<th>CLASS E</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN Calls</td>
<td>No. of Steps</td>
<td>Percent Deceived</td>
<td>Maximum Error</td>
</tr>
<tr>
<td>A1</td>
<td>55</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>A2</td>
<td>30</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>135</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>A4</td>
<td>30</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>A5</td>
<td>25</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 6**

Results for fourth order Runge-Kutta formula
### Table 7

Results for variable order block scheme

<table>
<thead>
<tr>
<th>CLASS A</th>
<th>FCN Calls</th>
<th>No. of Steps</th>
<th>Percent Deceived</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>21</td>
<td>14</td>
<td>0</td>
<td>0.570</td>
</tr>
<tr>
<td>A2</td>
<td>24</td>
<td>12</td>
<td>0</td>
<td>0.648</td>
</tr>
<tr>
<td>A3</td>
<td>117</td>
<td>58</td>
<td>0.05</td>
<td>1.218</td>
</tr>
<tr>
<td>A4</td>
<td>36</td>
<td>10</td>
<td>0</td>
<td>0.259</td>
</tr>
<tr>
<td>A5</td>
<td>36</td>
<td>12</td>
<td>0</td>
<td>0.084</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS B</th>
<th>FCN Calls</th>
<th>No. of Steps</th>
<th>Percent Deceived</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>216</td>
<td>112</td>
<td>0.02</td>
<td>1.296</td>
</tr>
<tr>
<td>B2</td>
<td>87</td>
<td>42</td>
<td>0</td>
<td>0.544</td>
</tr>
<tr>
<td>B3</td>
<td>36</td>
<td>21</td>
<td>0</td>
<td>0.244</td>
</tr>
<tr>
<td>B4</td>
<td>135</td>
<td>71</td>
<td>0</td>
<td>0.931</td>
</tr>
<tr>
<td>B5</td>
<td>126</td>
<td>45</td>
<td>0</td>
<td>0.517</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS D</th>
<th>FCN Calls</th>
<th>No. of Steps</th>
<th>Percent Deceived</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>75</td>
<td>49</td>
<td>0</td>
<td>0.979</td>
</tr>
<tr>
<td>D2</td>
<td>87</td>
<td>54</td>
<td>0.04</td>
<td>1.136</td>
</tr>
<tr>
<td>D3</td>
<td>126</td>
<td>68</td>
<td>0.03</td>
<td>1.224</td>
</tr>
<tr>
<td>D4</td>
<td>72</td>
<td>24</td>
<td>0</td>
<td>0.696</td>
</tr>
<tr>
<td>D5</td>
<td>102</td>
<td>33</td>
<td>0</td>
<td>0.552</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS E</th>
<th>FCN Calls</th>
<th>No. of Steps</th>
<th>Percent Deceived</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>48</td>
<td>32</td>
<td>0</td>
<td>0.608</td>
</tr>
<tr>
<td>E2</td>
<td>288</td>
<td>137</td>
<td>0</td>
<td>0.767</td>
</tr>
<tr>
<td>E3</td>
<td>126</td>
<td>63</td>
<td>0</td>
<td>0.529</td>
</tr>
<tr>
<td>E4</td>
<td>30</td>
<td>9</td>
<td>0</td>
<td>0.159</td>
</tr>
<tr>
<td>E5</td>
<td>36</td>
<td>8</td>
<td>0</td>
<td>0.218</td>
</tr>
</tbody>
</table>


