Efficient Integration for Continuous Simulation

J. C. Butcher, The University of Auckland

Abstract Within the enormous range of modelling situations involving time integration, we identify a number of special problem types that make great demands on the techniques of numerical analysis. These include stiff problems, retarded problems and problems involving algebraic constraints. We will review traditional numerical methods for these problems and attempt to identify the source of any special shortcomings that various methods might have for each class of problem. The discussion will then centre on singly-implicit methods and the potential that they have for dealing with these difficulties. The idea of a "General Linear Method" will be introduced as a means of including a very wide class of methods under a unified formulation. Finally, a new class of general linear methods will be discussed; this seems to have the potential to contribute in a useful way to the building up of a collection of tools for the solution of many difficult simulation problems.

1 INTRODUCTION

The modelling of continuous problems typically involves the integration of differential equations. This paper is concerned with the numerical approximation to the solution of of such equations or equation systems in the event that there is no closed-form or "analytic" solution. Since many problems involve some additional complication, such as the presence of stiffness, retarded terms or algebraic constraints, we will include these into the discussion as well.

The paper is organised along the lines of a brief survey of the problems themselves (Section 2), a survey of existing numerical methods (Section 3) and some suggestions for improving the numerical performance of numerical integrators by the use of some new approaches (Section 4). In the meantime we lay down some basic terminology.

We will always denote "time" by the symbol x and we consider the behaviour of a time dependent function $y(x)$. We will assume that the behaviour of y is governed by a differential equation

$$\frac{dy}{dx} = f(x, y). \tag{1}$$

Since (1) can have many solutions, it is necessary to specify some additional data to make the solution unique and to make numerical simulations feasible. We will deal here only with cases in which this is done using an "initial value" in which at some specified initial time x_0, y is assigned a specific value y_0. That is, we will assume that

$$y(x_0) = y_0. \tag{2}$$

The equation (1) is more general than it might seem at first sight because y could be a vector valued function with components y_1, y_2, ..., y_N each of which has a corresponding f component which might depend on each of the y components.

In this case, (1) is really a shorthand for an equation system of the form

$$\begin{array}{rcl} \frac{dy_1}{dx} &=& f_1(x, y_1, y_2, \ldots, y_N), \\ \frac{dy_2}{dx} &=& f_2(x, y_1, y_2, \ldots, y_N), \\ \vdots & & \vdots \\ \frac{dy_N}{dx} &=& f_N(x, y_1, y_2, \ldots, y_N). \end{array} \tag{3}$$

With such generality as this, it is possible to include within the formulation high order differential equations. If the generality is apparently lessened by removing the dependence of each of f_1, f_2, ..., f_N on x, so that the equation is assumed to be "autonomous", then this makes no essential difference because an additional differential equation whose solution is exactly x can be added to the system. Thus we can consider instead of (3) the simpler system

$$\begin{array}{rcl} \frac{dy_1}{dx} &=& f_1(y_1, y_2, \ldots, y_N), \\ \frac{dy_2}{dx} &=& f_2(y_1, y_2, \ldots, y_N), \\ \vdots & & \vdots \\ \frac{dy_N}{dx} &=& f_N(y_1, y_2, \ldots, y_N). \end{array} \tag{4}$$

In the use of differential equation software, (3) is preferred, because of its convenience, whereas for some mathematical analyses, (4) is preferred because of its greater simplicity with no less generality.

2 TIME DEPENDENT PROBLEMS

In this section we will explore some special problem classes which either place abnormal difficulties on a numerical process or which extend

the problem class to such an extent that completely new methods become necessary.

2.1 Stiff differential equations

Many differential equations, and equation systems, are very strongly stable in the sense that a small permutation from the solution will be quickly attenuated. For example, the initial value problem

$$\frac{dy}{dx} = -1000y + 999e^{-x}, \qquad y(0) = 1, \qquad (5)$$

has solution $y = e^{-x}$. If the initial value is changed from 1.0 to 1.001 then by the time x moves from 0.0 to 0.01, the solution will have changed from its original value of $e^{-0.01} = 0.9900498337$ to 0.9900498791. Hence the perturbation of 0.001 has, after this small time interval, been decreased to 4.54×10^{-8}. If x increases further to 0.1, the effect of the perturbation will have decreased to less than 10^{-46}. This same problem, which is very stable as concerns the exact solution, is actually very unstable when a standard numerical method is attempted for its numerical approximation. To see how this happens, consider what happens when numerical approximations are calculated using the simple method of Euler. For this method, the solution approximation in each time step is assumed to change linearly with its slope evaluated at the beginning of the step. For example, assume that steps of size $h = 0.01$ are used in approximating the solution. The exact solution at $x = 1.0$ is 0.3679. When the simpler problem $dy/dx = -y$, with the same solution, is solved by the Euler method, the computed answer is 0.3660 which is correct to about 0.5%. However, when the same computation is attempted for (5) no useful result is found. For example after 10 steps, when x has moved to 0.1, the solution has now increased out of hand to nearly 7 million. By the time x has increased to 1.0 the computed y has increased to more than 5×10^{92}. This instability results from no perturbation at all, except the errors that arise naturally in the numerical approximation.

This example of a "stiff" problem is simple but typical of stiff behaviour in general. Problems with this particular difficult feature arise in the modelling many physical, chemical and engineering problems.

2.2 Delay differential equation systems

Modelling using differential alone is not adequate in the description of many economic and biological problems because the rate of change of the solution will depend on the state of the system, not only at the current time but also on the solution at some previous time, or even on several previous times. The so-called logistic equation

$$\frac{dy}{dx} = Ry(x)(1 - y(x)),$$

is used in the study of populations that grow exponentially with rate factor R for small population sizes, but for which the exponential growth factor decreases as the population increases towards a maximum possible size, scaled to 1 in this formulation. An alternative delay version of the same problem assumes that the growth rate is related not to the current population size, but to the population at some previous time. For example, taking the unit of delay as 1 gives the equation

$$\frac{dy}{dx} = Ry(x)(1 - y(x - 1)).$$

There are special difficulties associated with this sort of problem related to the fact that evaluation of the delay term involves interpolation within steps previously covered. Discontinuous behaviour is also characteristic of the solutions of this type of problem and this also creates numerical difficulties.

2.3 Differential-algebraic equation systems

Sometimes additional algebraic constraints have to be added to a differential equation system to render the system a faithful model of the scientific problem being investigated. This can be interpreted, in many physical situations, as restraining the solution to a specific subspace of the N dimensional vector space in which the solution of a differential equation lies. For many other problems, the algebraic constraints can be interpreted as the imposition of conservation laws. Many differential-algebraic equations can be solved using a simple extension of methods appropriate for differential equations. However, there are also problems for which compeletely new numerical methods are necessary. An example of this is in the formulation of the motion of a simple pendulum using X, Y to represent the position coordinates, U and V to represent the corresponding velocity coordinates and T for the tension in the "light string". Even though we do not have a differential equation for T, we know that its value is always exactly the required value to force the length of the string, here denoted by L, to remain constant. The acceleration due to gravity will be denoted, as usual, by g and the mass of the moving bob by m. The complete set of differential and algebraic equations to model the problem is

$$\frac{dX}{dx} = U,$$

$$\begin{aligned}
\frac{dY}{dx} &= V, \\
\frac{dU}{dx} &= -\frac{TX}{Lm}, \\
\frac{dV}{dx} &= g - \frac{TY}{Lm}, \\
X^2 + Y^2 &= L^2.
\end{aligned}$$

This problem is typical of mechanical problems and is said to have "index 3". The index measures how far removed the problem is from being just a differential equation and at the same time measures the inherent computational difficulties there are associated with its solution. Index 3 is on the limits of what is possible to be solved numerically in any satisfactory manner.

3 TRADITIONAL METHODS

In a computational situation, being presented with an initial value problem is equivalent to being presented with a procedure or subroutine to compute values of the vector-valued function f and, at the same time, sufficient inital data to specify the solution. By using the function f repeatedly, the aim is to advance what is known about the solution forward in time. The traditional approach is to carry this out in a step-by-stp manner. That is, once the initial information has been organised into a suitable form, the solution is advanced one step forward and the re-organised initial information is updated to apply one step later. The simple method of Euler, which we have already discussed, can be written in the form

$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1}), \qquad (6)$$

where h is the steplength equal to $x_n - x_{n-1}$ and y_n is the computed approximation to $y(x_n)$, with y_0 the given initial value. Since each of y_0, y_1, ... is a vector, we have already reached a notational crisis by trying to write the components of the sequence of approximations separately. We will do this just once, for the Euler method, and from then on confine ourselves to the more compact vector notation. Component i of the approximation to $y(x_n)$ will be denoted by y_{ni}. This means that (6) can be written in full as

$$y_{ni} = y_{n-1,i} + hf_i(x_{n-1}, y_{n-1,1}, \ldots, y_{n-1,N}), \quad i = 1, 2, \ldots, N. \qquad (7)$$

3.1 Runge-Kutta methods

Runge-Kutta methods are "one-step methods" in the sense that, as for the Euler method, the formula for y_n involves y_{n-1}, *but none of the earlier approximations* y_{n-2}, y_{n-3}, ... This means that no preparation is required before the very first step is carried out and no special provision is required if for some reason the stepsize h is altered between one step and the next. Unlike the Euler method, f can be evaluated more than once in each step and the approximations at which these derivtive values are computed are known as "stage-values". In an s-stage method, s of these stage values, Y_1, Y_2, ..., Y_s are evaluated in each step. In a classical "explicit" form of the method, each stage depends on the previously computed stage derivatives, which we will denote by F_1, F_2, ..., F_s. Suppose that the coefficient of hF_j in the formula for Y_i is a_{ij} and suppose also that Y_i is intended to be an approximation to $y(x_{n-1} + hc_i)$. Then it is necessary for consistency between possible interpretations of the method that $c_i = \sum_{j=1}^{s} a_{ij}$. Because of the explicit nature of the method, $a_{ij} = 0$ unless $i > j$, and a consequence of this is that $c_1 = 0$. Write b_i for the coefficient of hF_i in the approximation to y_n output at the end of the step. Put all this together and we find the following formulae for carrying out the work in a single step

$$\begin{aligned}
Y_i &= y_{n-1} + h\textstyle\sum_{j=1}^{s} a_{ij}F_j, \\
F_i &= f(x_{n-1} + hc_i, Y_i), \qquad i = 1, 2, \ldots, s, \\
y_n &= y_{n-1} + h\textstyle\sum_{i=1}^{s} b_iF_i.
\end{aligned}$$

It is customary to write the defining coeffients for a particular method in a tableau as follows

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}$$

or in the explicit case, when many of the coefficients are zero,

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}$$

The most famous example of these methods was discovered by Kutta and is often referred to as *the* Runge-Kutta method. In this case the tableau is

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

In contrast to these explicit methods, methods also exist in which the rule that certain entries

in the A matrix must be zero, is not enforced. These implicit methods can, in some cases, be used for the solution of stiff problems, unlike explicit methods where they never works efficiently. An example of one of these implicit methods is given by the tableau

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

3.2 Linear multistep methods

Unlike Runge-Kutta methods which depend on repeated calculation of f to obtain good accuracy, linear multistep methods evaluate f only once in each step. However, unlike Runge-Kutta methods, the computed solution in each step may depend on values computed in several previous steps. Denote by k the greatest distance back in the past on which each approximation depends. That is, y_n may depend on y_{n-k} but on no earlier value. Since we can compute $f(x_i, y_i)$ as soon as y_i has been evaluated, the general form of one of these "linear k-step methods" is

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f(x_{n-i}, y_{n-i}).$$

Note that methods in this class are explicit when $\beta_0 = 0$. The alternative implicit methods, in which $\beta_0 \neq 0$, are capable of greater accuracy but suffer the same disadvantages as implicit Runge-Kutta methods, of requiring special techniques to actually evaluate solution values. Examples of each type are easy to find and they are often combined for practical implementation into "predictor-corrector" pairs. For convenience we will write f_i to mean $f(x_i, y_i)$ Such a pair, known as order 3 Adams-Bashforth and Adams-Moulton methods are given by the formulae

$$\begin{aligned} y_n &= y_{n-1} + h\left(\frac{23}{12}f_{n-1} - \frac{4}{3}f_{n-2} + \frac{5}{12}f_{n-3}\right), \\ y_n &= y_{n-1} + h\left(\frac{5}{12}f_n + \frac{2}{3}f_{n-1} - \frac{1}{12}f_{n-2}\right). \end{aligned}$$

The explicit (predictor) member of the pair can be used to obtain an approximation good enough to evaluate f_n for use in the implicit (corrector) member of the pair.

By contrast to these methods which work well for non-stiff problems, there are also methods that are recommended for stiff problems. The most important of these are known as "backward difference methods" and the 3-step example is

$$y_n = \frac{18}{11}y_{n-1} - \frac{9}{11}y_{n-2} + \frac{2}{11}y_{n-3} + h\frac{6}{11}f_n.$$

3.3 Taylor series methods

For many differential equations, it is possible to derive formulae also for the second and higher derivatives and thus it is possible to compute several terms in the Taylor series approximation for $y(x_n)$ in terms of $y(x_{n-1})$, $y'(x_{n-1})$, $y''(x_{n-1})$, ... Even when formulae for the higher derivatives cannot be found by inspection, computational methods exist for generating computer code for them.

3.4 Critique of existing methods

Although the methods that have been referred to have enjoyed great success for a variety of problems, there are reasons to wish for more. For the solution of non-stiff problems that impose no special difficulties, Runge-Kutta methods are regarded as being more expensive to use although they enjoy the advantages of good stability and convenience of use. There is no built-in interpolation scheme or asymptotically correct local error estimator and this makes it difficult to design flexible, variable step algorithms based on these methods. On the other hand linear multistep methods have interpolation and error estimation readily available but variable step implementations are complicated by the need to modify the way the data is used when stepsizes actually change. Taylor series methods are difficult to implement for general purpose use, and although they can be spectacularly successful, they have never enjoyed the popularity of either Runge-Kutta or linear multistep methods.

Stiff problems are usually solved using backward difference methods but stability difficulties can limit the accuracy that is achievable by these methods. Two key indicators of the ability of a method to solve stiff problems accurately and efficiently are (i) order of accuracy which measures how rapidly accuracy improves when increasingly severe stepsize limitations are imposed and (ii) A-stability which determines the ability of method to solve stable problems and produce stable numerical results. It is known that A-stability is impossible for orders greater than 2. The same limitations do not apply to implicit Runge-Kutta methods and it is actually possible to obtain A-stable methods of any order. The trouble with these methods is, however, that the computational cost rises rapidly with order. In recent times, Runge-Kutta methods have been specifically sought which, to some extent, avoid the high costs associated with fully implicit methods. In section 4 we will explore these specially designed methods in some detail. The difficulties encountered in this search for good methods has also led to other considerations. Perhaps, it

might be thought, it is possible to look beyond the standard types of methods for a completely new approach. Fortunately, such a new approach is possible and we will look for it within the wide class of "general linear methods", which includes Runge-Kutta methods and linear multistep methods as special cases.

4 SOME NEW APPROACHES

4.1 General linear methods

As a unifying scheme for representing a large family of methods, we look at methods with r quantities passed between steps and with s stages. Such a method can be represented by a partitioned $(s+r)\times(s+r)$ matrix

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right]$$

Denote the quantities passed from the completion of step number n to the following step by $y_1^{[n]}$, $y_2^{[n]}$, ..., $y_r^{[n]}$ and the stage values by Y_1, Y_2, ..., Y_s; with F_1, F_2, ..., F_s the corresponding stage derivatives. These quantities are related by the equations

$$\begin{aligned} Y_i &= h\textstyle\sum_{j=1}^{s} a_{ij}F_j + \sum_{j=1}^{r} u_{ij}y_j^{[n-1]}, \quad j=1,\ldots,s,\\ y_i^{[n]} &= h\textstyle\sum_{j=1}^{s} b_{ij}F_j + \sum_{j=1}^{r} v_{ij}y_j^{[n-1]}, \quad j=1,\ldots,r. \end{aligned}$$

By choosing $r=1$ we revert to the Runge-Kutta case and by choosing $s=1$ we obtain methods that are similar to linear multistep methods. Our aim will be to find methods that lie between these extreme cases and have properties that are, overall, superior to either of them. We will write p for the order of the method and q for the "stage order". To see what this means, suppose that there exist r functions ϕ_1, ϕ_2, ..., ϕ_r such that the method accurately propagates a vector of approximations formed from the exact trajectory by applying these functions. If $\phi(y(x))$ is the vector of approximations formed in this way then the criteria for order p and stage order q is that when $y_i^{[n-1]}$ is defined for $i=1,2,\ldots,r$ as $\phi_i(y(x_{n-1}))$, then stage number i satisfies $Y_i = y(x_{n-1}+hc_i)+O(h^{q+1})$, for $i=1,2,\ldots,s$ and $y_i^{[n]} = \phi(y(x_n))+O(h^{p+1})$, for $i=1,2,\ldots,r$. Abstract though this definition might seem, it can be turned into a practical means of identifying useful methods as we shall see later.

4.2 Diagonally implicit Runge-Kutta methods

A well established approach to the selection of suitable Runge-Kutta methods for stiff problems is to insist that the coefficient matrix A has a lower triangular structure with constant values on the diagonal. Even though this method is still implicit, the stages can be evaluated sequentially. This means that rather than having to solve a system of sN non-linear algebraic equations in each time step it is necessary to solve only s systems each of N equations. This is a tremendous saving for the large problems that occur in practice. Unfortunately the methods found under this restriction are limited in their usefulness by low stage order and by an excessive number of stages required for reasonably high orders. It will be seen below that the same diagonally implicit structure does not have these same limitations in the context of general linear methods.

4.3 Singly-implicit Runge-Kutta methods

To overcome the disadvantages of diagonally-implicit methods, without losing all their advantages, singly implicit methods have come under consideration. These methods are fully implicit but the coefficient matrix A has a one-point spectrum; that is, it has only a single eigenvalue. By adding special transformations within the implementation, the cost can be reduced, at least for large problems, to essentially the same as for diagonally-implicit methods. It is also possible to obtain order and stage order each equal to s. Unfortunately, additional difficulties are introduced in this type of method. Most notably, the choice of abscissae for the method required for A-stability forces some of them to lie outside the interval over which a step is working, if $s \geq 3$. Two approaches can be adopted to alleviate this difficulty and they can be combined together for a combined benefit. The first of these allows for the addition of further diagonally-implicit stages tagged onto the end of the main singly implicit block. The second is to generalise the order requirement to what is known as "effective order". This is similar to the meaning of order adopted for general linear methods and does not result in any deterioration of the computational performance.

4.4 DIMSIM methods

As an attempt to identify promising general linear methods, DIMSIM or "diagonally implicit multistage integartion methods" were introduced. The basic assumptions were that $p \approx q \approx r \approx s$, that A has a diagonally implicit structure and that V has rank 1. Although this class contains methods intended for parallel computation, we shall not discuss these here and there remains what ar called "type 1" and "type 2" methods, intended for non-stiff and stiff problems respectively. To simplify the stability analysis of these methods, it is assumed that methods of these

types possess a property known as RK stability, which is simply the requirement that the stability matrix for the method has only a single non-zero eigenvalue. Methods of this sort for the explicit type 1 variety and the implicit type 2 variety up to quite high orders but they become increasingly difficult to actually find. So far methods up to order 8 are known, at least with numerically derived coefficients.

4.5 A new class of methods

It has been discovered recently that "inherent RK stability is possible". That is, it is possible to impose some mild restrictions on the structure of the methods so that good stability, just as for Runge-Kutta methods is assured. Furthermore, unlike DIMSIM methods, the coefficients in these methods can be evaluated exactly, using rational operations. We will devote most of our attention to the implicit members of this new class. We recall from 4.1 that for a method to be of order p, it is necessary that however the approximations $y_i^{[n-1]}$ are derived from $y(x_{n-1})$, the result at the end of a step should agree with similar quantities derived from $y(x_n)$ to within $O(h^{p+1})$. If $r = q = p$, as we will assume, then the relationship between these quantities is necessarily of the form

$$y_i^{[n]} = \sum_{j=0}^{p} \alpha_{ij} h^j y^{(j)}(x_n) + O(h^{p+1}), \quad i = 1, 2, \ldots, r.$$

For methods with inherent Runge-Kutta stability, the matrix of α coefficients has the simple form

$$\begin{bmatrix} 1 & \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ 0 & 1 & \alpha_1 & \cdots & \alpha_{p-1} \\ 0 & 0 & 1 & \cdots & \alpha_{p-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Under these assumptions, V is necessarily of the

$$V = \begin{bmatrix} 1 & v_{12} & v_{13} & \cdots & v_{1,p+1} \\ 0 & v_{22} & v_{23} & \cdots & v_{2,p+1} \\ 0 & v_{32} & v_{33} & \cdots & v_{3,p+1} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & v_{p+1,2} & v_{p+1,3} & \cdots & v_{p+1,p+1} \end{bmatrix}$$

and we will assume that the eigenvalues of this matrix are $\{1, 0, 0, \ldots, 0\}$ and, as the final requirement for inherent Runge-Kutta stability that the matrices

$$BA - JB \quad \text{and} \quad BU - JV + VJ$$

are zero except for their first rows, where

$$J = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

It can be shown that the stability matrix for a method with these properties is

$$M = (I - zJ)\widehat{V}(I - zJ)^{-1},$$

where $\widehat{V}$ is identical to V except for the first row. This implies that the method has Runge-Kutta stability.

It turns out to be possible to construct methods with these properties and they have excellent prospects as stiff solvers. A related type of method discovered by an Auckland research student, William Wright, has similar properties but is explicit and is suitable for the solution of non-stiff problems.

5 Acknowledgements

The author acknowledges the support of the Marsden Fund of New Zealand. He has had numerous and valuable conversations with members of the weekly Auckland Numerical Analysis workshop and particularly with Shirley Huang and Will Wright each of whom is working on numerical methods with inherent Runge-Kutta stability.

References

Alexander, R., Diagonally implicit Runge-Kutta methods for stiff ODEs, *SIAM J. Numer. Anal.* 14 (1977), 1006–1021.

Barton, D., I. M. Willers and R. V. M. Zahar, The automatic solution of ordinary differential equations by the method of Taylor series, *Comput. J.* 14 (1971), 243–248.

Bashforth, F. and J. C. Adams, An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid, with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops, Cambridge University Press, Cambridge, (1883).

Burrage, K., A special family of Runge-Kutta methods for solving stiff differential equations, *BIT*, 18 (1978), 22–41.

Burrage, K., J. C. Butcher and F. H. Chipman, An implementation of singly-implicit Runge-Kutta methods, (with K. Burrage, F. Chipman), *BIT* 20 (1980), 326-340.

Butcher, J. C., Implicit Runge-Kutta processes, *Math. Comp.* 18 (1964), 50-64.

Butcher, J. C., On the convergence of numerical solutions of ordinary differential equations, *Math. Comp.* 20 (1966), 1-10.

Butcher, J. C., The effective order of Runge-Kutta methods, it Lecture Notes in Mathematics 109, (1969), 133-139.

Butcher, J. C., The order of numerical methods for ordinary differential equations, *Math. Comp.* 27 (1973), 793-806.

Butcher, J. C., On the implementation of implicit Runge-Kutta methods, *BIT* 16 (1976), 237-240.

Butcher, J. C., A transformed implicit Runge-Kutta method, *J. Assoc. Comput. Mach.* 26 (1979), 731-738.

Butcher, J. C., General linear methods: a survey, *Appl. Numer. Math.* 1 (1985) , 273-284.

Butcher, J. C., Towards efficient implementation of singly-implicit methods, *ACM Transactions Math. Software* 14 (1988), 68-75.

Butcher, J. C. and J. Cash, Towards efficient Runge-Kutta methods for stiff systems, *SIAM J. Numer. Anal.* 27 (1990) 753-761.

Butcher, J. C., Diagonally-implicit multi-stage integration methods, *Appl. Numer. Math.* 11 (1993), 347-363.

Butcher, J. C., An introduction to DIMSIMs, *Comp. Appl. Math.*, 14 (1) (1995), 59- 72.

Butcher, J. C., General linear methods, *Comput. Math. Appl.* 31 (1996), 105-112.

Butcher, J. C. and Z. Jackiewicz, Construction of diagonally implicit general linear methods of type 1 and 2 for ordinary differential equations, *Appl. Numer. Math.* 21 (1996), 385-415.

Butcher, J. C., J. R. Cash and M. T. Diamantakis, DESI methods for stiff initial value problems, *ACM Trans. Math. Software* 22 (1996), 401-422.

Butcher, J. C. and P. Chartier, A generalization of singly-implicit Runge-Kutta methods, *Appl. Numer. Math.* 24 (1997), 343-350.

Butcher, J. C., Z. Jackiewicz and H. D. Mittelmann, A nonlinear optimization approach to the construction of general linear methods of high order, *J. Comput. Appl. Math.* 81 (1997), 181-196.

Butcher, J. C. and Z. Jackiewicz, Implementation of diagonally implicit multistage integration methods for ordinary differential equations, *SIAM J. Numer. Anal.* 34 (1997), 2119-2141.

Butcher, J. C., P. Chartier and Z. Jackiewicz, Nordsieck representation of DIMSIMs, *Numer. Algorithms* 16 (1997), 209-230.

Butcher, J. C. and Z. Jackiewicz, Construction of high order diagonally implicit multistage integration methods for ordinary differential equations, *Appl. Numer. Math.* 27 (1998), 1-12.

Butcher, J. C. and M.T. Diamantakis, DESIRE: diagonally extended singly implicit Runge-Kutta effective order methods, *Numer. Algorithms*, 17 (1998), 121-145

Butcher, J. C. and P. Chartier, The effective order of singly-implicit Runge-Kutta methods, *Numer. Algorithms*, 20 (1999), 269-284.

Calahan, D. A., A stable, accurate method of numerical integration for nonlinear systems, *Proc. IEEE* 56, (1968), 744.

Curtiss,C. F. and J. O. Hirschfelder, Integration of stiff equations, *Proc. Nat. Acad. Sci.* 38 (1952), 235–243.

Dahlquist, G. A special stability problem for linear multistep methods. *Nordisk Tidskr. Informations-Behandling* 3 (1963) 27–43.

Hairer, E., C. Lubich and M. Roche, The numerical solution of differential-algebraic systems by Runge-Kutta methods, *Lecture Notes in Mathematics, 1409* Springer-Verlag, (1989).

Kutta, W., Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, *Z. Math. Phys.* 46 (1901), 435–453.

Moulton, F. R., *New methods in exterior balistics*, University of Chicago, (1926).

Runge, C., Über die numerische Auflösung von Differentialgleichungen, *Math. Ann.* 46 (1895), 167–178.