On the computational aspects of semi-implicit Runge-Kutta methods

J. R. Cash and C. B. Liem*

Department of Mathematics, Imperial College of Science and Technology, Queen's Gate, London SW7 2BZ

In recent years the problem of obtaining an approximate numerical solution of stiff systems of first order ordinary differential equations has received a great deal of attention. One important class of numerical procedures suitable for tackling this problem is the class of semi-implicit Runge-Kutta procedures originally proposed by Rosenbrock. Although a great many algorithms belonging to this particular class have now been proposed a detailed comparison between them on a truly representative class of stiff systems has not so far been made. Moreover an analysis of the computational aspects of semi-implicit Runge-Kutta schemes is noticeable only by its absence. The purpose of the present paper is to examine some of these computational aspects in the third order case in some detail with the aim of making some specific recommendations regarding which methods should be used in practice. The details of some fairly exhaustive numerical experiments are also presented.

(Received May 1977)

1. Introduction

In the present paper we shall be concerned with the approximate numerical integration of stiff systems of first order ordinary differential equations of the form

$$\frac{dx}{dt} = f(x)\,,\ x(t_0) = x_0 \tag{1.1}$$

In an attempt to derive a class of Runge-Kutta procedures suitable for the numerical integration of (1.1) Rosenbrock (1963) proposed the class of semi-implicit methods given by

$$x_{n+1} - x_n = h\sum_{r=1}^{R} c_r k_r$$

$$k_1 = f(x_n) + \alpha_1 h \frac{\partial f}{\partial x}(x_n)k_1 \tag{1.2}$$

$$k_r = f\left(x_n + h\sum_{s=1}^{r-1} b_{rs}k_s\right) + \alpha_r h \frac{\partial f}{\partial x}\left(x_n + h\sum_{s=1}^{r-1}\beta_{rs}k_s\right)_{k_r}$$

$$r = 2,3,\ldots,R$$

where the c_i's, α_i's, β_{ij}'s and b_{ij}'s are constants. Several algorithms of this particular class have more recently been proposed (2, 3, 4, 5) but because of the inherent difficulty in deriving high order schemes (see particularly Calahan, 1968) attention has been mainly focussed on the problem of actually obtaining schemes rather than on an examination of the computational aspects of existing methods. The main practical problem associated with the use of (1.2) lies in the estimation of the local truncation error. An algorithm containing a built-in error estimate of the type proposed by Merson (1957) has been derived by Haines (1969) but no real examination of its computational efficiency seems to have been carried out so far. The present paper arose as a result of some numerical experiments performed with Haines' method which in some cases was found to produce rather unexpected results. An examination of the theoretical aspects of Haines' algorithm by the second author (CBL) revealed that the expression for the characteristic root given by Haines is incorrect. It was found that the correct expression for the characteristic root is given by

$$(1 - 8q/3 + (11q^2)/6 + q^3/3)/(1 - 11q/3 + 5q^2 - 3q^3 + (2q^4)/3),\ q = h\lambda \tag{1.3}$$

The third coefficient in the numerator of (1.3), (= 11/6), is different from 2/9 which is the corresponding coefficient in the expression for the characteristic root given by Haines.

*Now at Department of Mathematics, Hong Kong, Polytechnic.

This error also invalidates Haines' plot for the region of absolute stability of his method (although the true region is not very different from his) and for a consideration of this and other theoretical aspects of semi-implicit Runge-Kutta methods the reader is referred to Liem (1976). The fact that this error has remained undetected so far and has been duplicated several times in the literature would seem to support our claim that a detailed analysis of the computational aspects of semi-implicit Runge-Kutta methods has not so far been carried out. In view of this we decided to make some comparisons between four specific third order methods to see what general conclusions could be drawn. In particular we wanted to see how the error estimation procedure proposed by Haines, which is strictly valid only for linear systems, performed on general nonlinear problems and we also wanted to see how expensive Haines' method is in terms of computing time compared with certain other methods. By carrying out such an analysis we were able to gain some important information regarding error estimates of this type since, although our attention was confined to the third order case in this paper, it is expected that most of our general conclusions carry over to higher order equations as well.

The methods which we have chosen for comparison and the corresponding test problems are given in Section 2 and a summary of the results together with the main recommendations are given in Section 3.

2. Some third order methods

Since the only serious attempt to derive a semi-implicit Runge-Kutta procedure of the form (1.2) containing a built-in error estimate has been made by Haines, and Haines' method is third order, it seemed appropriate to carry out our investigation by considering some third order methods. The schemes which we chose for comparison purposes were as follows:

1. Haines' method.
2. Calahan's (1968) method which is given by

$$x_{n+1} - x_n = 3/4k_1 + 1/4k_2$$

$$k_1 = h(I - ha\frac{\partial f}{\partial x}(x_n))^{-1}f(x_n)$$

$$k_2 = h(I - ha\frac{\partial f}{\partial x}(x_n))^{-1}f(x_n + bk_1)$$

where $a = 0{\cdot}788675134$, $b = -1{\cdot}15470054$, and two methods of the general form

$$x_{n+1} - x_n = c_1k_1 + c_2k_2 + c_3k_3$$
$$k_1 = h(I - a_1hA(x_n))^{-1}f(x_n)$$
$$k_2 = h(I - a_2hA(x_n + e_1k_1))^{-1}f(x_n + b_1k_1)$$
$$k_3 = h(I - a_3hA(x_n + e_2k_1 + e_3k_2))^{-1}f(x_n + b_2k_1 + d_1k_2)$$

where $A(x) = \frac{\partial f}{\partial x}(x)$. The two sets of coefficients which we choose are

3. $c_1 = 13/4$, $c_2 = 3/4$, $c_3 = -3$, $a_1 = a_2 = 1/2$, $b_1 = -2/3$, $e_1 = e_2 = e_3 = b_2 = d_1 = 0$, $a_3 = 1/3$

and

4. $c_1 = 2/3$, $c_2 = 0{\cdot}1345999274$, $c_3 = 0{\cdot}1987334059$, $a_1 = a_2 = a_3 = 0{\cdot}4358665215$, $b_1 = -1$, $b_2 = 0{\cdot}6013743641$, $d_1 = 0{\cdot}3986256359$, $e_1 = e_2 = e_3 = 0$. (see Cash, 1976).

The expressions for the characteristic roots of 3. and 4., denoted by $R_1(h\lambda)$ and $R_2(h\lambda)$ respectively, are given by

$$R_1(h\lambda) = (1 - q/3 - q^2/4)/(1 - 4q/3 + 7q^2/12 - q^3/12), \quad q = h\lambda$$
$$R_2(h\lambda) = (1 - (3a - 1)q + (3a^2 - 3a + 1/2)q^2)/(1 - 3aq + 3a^2q^2 - a^3q^3),$$

where $a = 0{\cdot}4358665215$. It can easily be shown that both of these schemes are L-stable and have order 3. Associated with schemes 2, 3 and 4 we use an '$h - 2h$' error estimation procedure coupled with local Richardson extrapolation and we shall explain this in more detail later on. If we now take two steplengths of integration as our unit of comparison we find that in integrating two equal steps the following main computations are required:

Method	*Function evaluations*	*Jacobian evaluations*	*Matrix inversions*
1	10	6	10
2	5	2	3
3	5	2	6
4	8	2	3

We see immediately from this table that Haines' method requires more function evaluations, more Jacobian evaluations and more matrix inversions than any of the other methods and as a result we would immediately question its efficiency. In general it would seem that scheme 2 requires the least computational effort but it suffers from the disadvantage of not being L-stable and very poor results have been obtained on excessively stiff problems. Unsatisfactory results when using Calahan's method for certain problems have also been reported by Lapidus and Seinfeld (1971). We chose method 3 so as to keep the number of function evaluations to a minimum and method 4 was chosen to keep the number of Jacobian evaluations and matrix factorisations to a minimum. For a detailed derivation of 4 the reader is referred to Liem (1976).

We now describe our step control procedure in more detail. Rewriting our semi-implicit Runge-Kutta procedure in the form

$$x_{n+1} - x_n = h\phi(x_n,h)$$

it follows that the local truncation error of our scheme is given by

$$T_{n+1} = x(t_{n+1}) - x(t_n) - h\phi(x(t_n),h)$$

where $x(t_n)$ is the theoretical solution of our initial value problem at the step point t_n. Assuming that $x(t)$ is sufficiently differentiable we obtain a relation of the form

$$T_{n+1} = \psi(x(t_n))h^4 + 0(h^5)$$

where $h^4\psi(x)$ is the principal local truncation error of our scheme. If now starting from the point (t_n,x_n) we use two different steplengths h and $2h$ to get two distinct approximations to $x(t_{n+2})$ we may use these two approximations to give an estimate of the principal local truncation error. Firstly if we use a steplength $2h$ to get our first approximation x^*_{n+2} to $x(t_{n+2})$ we have

$$x(t_{n+2}) - x^*_{n+2} = T^*_{n+1} = (2h)^4\psi(x(t_n)) + 0(h^5).$$

If we now compute x_{n+2}, the second approximation to $x(t_{n+2})$, by applying our method twice with a steplength h we have

$$x(t_{n+2}) - x_{n+2} = T_{n+1} = 2h^4\psi(x(t_n)) + 0(h^5).$$

Subtracting these two relations and ignoring the terms $0(h^5)$ we have

$$\varepsilon_{n+2} \equiv x(t_{n+2}) - x_{n+2} \sim 1/7(x_{n+2} - x^*_{n+2}) \qquad (2.1)$$

and this serves as a computable approximation to the local truncation error of our scheme used with a steplength h. When dealing with stiff systems it can of course be dangerous to ignore the $0(h^5)$ terms since the values of the derivatives included in these terms may be very large. Practical experience has however shown that (2.1) is usually a reasonable estimate of the local truncation error. We may use this error estimate to control the stepsize in the following way:

Let $\theta(t)$ be the maximum local truncation error allowed in the numerical solution when integrating from t to the next step point. After every two steps a comparison between the estimated local truncation error and the maximum allowable error $\theta_n \equiv \theta(t_n) + \theta(t_{n+1})$ is made and the new steplength is determined. If

1. $\max_j|\varepsilon_{n+2}|_j > \theta_n$ halve h and return to the grid point (t_n,x_n).
   Here $(\varepsilon_{n+2})_j \equiv j$th component of ε_{n+2}.
2. $\max_j|\varepsilon_{n+2}|_j < \theta_n/25$ double h and continue from (t_{n+2},x_{n+2}).

Table 1 **Results for problem C1**

Method	*Function calls*	*Jacobian calls*	*Inversion calls*	*Number of steps*
1	1892	1118	1935	386
2	745	285	460	285
3	594	227	734	227
4	995	231	382	231

$F_1:F_3 = 3{\cdot}2, J_1:J_3 = 5, I_1:I_3 = 2{\cdot}6, N_1:N_3 = 1{\cdot}7$

Table 2 **Results for problem C2**

Method	*Function calls*	*Jacobian calls*	*Inversion calls*	*Number of steps*
1	2184	1308	2190	437
2	823	324	499	324
3	648	254	788	254
4	1082	264	409	264

$F_1:F_3 = 3{\cdot}4, J_1:J_3 = 5{\cdot}1, I_1:I_3 = 2{\cdot}8, N_1:N_3 = 1{\cdot}7$

Table 3 **Results for problem D2**

Method	*Function calls*	*Jacobian calls*	*Inversion calls*	*Number of steps*
1	13033	7820	13038	2604
2	1817	724	1093	724
3	1397	556	1682	556
4	2966	738	1114	738

$F_1:F_3 = 9{\cdot}3, J_1:J_3 = 14, I_1:I_3 = 7{\cdot}8, N_1:N_3 = 4{\cdot}7$

Table 4 Results for problem D5

Method	*Function calls*	*Jacobian calls*	*Inversion calls*	*Number of steps*
1	1018	608	1025	204
2	308	118	190	118
3	268	102	332	102
4	434	102	166	102

$F_1:F_3 = 3{\cdot}8,\ J_1:J_3 = 6,\ I_1:I_3 = 3{\cdot}1,\ N_1:N_3 = 2.$

Table 5 Results for problem E5

Method	*Function calls*	*Jacobian calls*	*Inversion calls*	*Number of steps*
1	14919	8947	14930	2985
2	2261	898	1363	898
3	1948	774	2348	774
4	4066	1010	1528	1010

$F_1:F_3 = 7{\cdot}7,\ J_1:J_3 = 11{\cdot}6,\ I_1:I_3 = 6{\cdot}4,\ N_1:N_3 = 3{\cdot}9$

3. Otherwise keep h fixed and continue from (t_{n+2},x_{n+2}).

One of the major practical difficulties in making a numerical comparison of integration methods is that of choosing a truly representative class of test problems. This difficulty has, however, now been largely removed for small systems at least as a result of a recent paper by Enright *et al* (1975) which lists an excellent selection of suitable test problems. For the purposes of the analysis presented in this paper we did not think it worthwhile to consider any linear problems, since most A-stable methods will perform fairly well on this particular class of problem, and so we confine our attention to the nonlinear problems C1, C2, D2, D5 and E5 which may be found, together with their original sources, in Enright *et al* (1975).

3. Conclusions and recommendations

3.1. *Comparison of accuracy*

In order to make a comparison between the degrees of accuracy obtained at the end points of the range of integration the problems were run first of all with $\theta(t) = \theta = 10^{-4}$ and then with $\theta = 10^{-6}$. We concede that different conclusions may possibly have been obtained if different tolerances (and in particular if a variable tolerance such as the one suggested by Enright *et al*) had been used but in any comparison certain constraints must necessarily be imposed to keep the amount of information generated to a manageable size and one of our constraints was that fixed tolerances were used. It was found that with a given fixed error tolerance almost exactly the same degree of accuracy was obtained for all methods leading to the conclusion that there is little to choose between them in this respect.

3.2. *Comparison of computational effort*

The operations which need to be counted in general for the purposes of our comparison are the numbers of function evaluations, Jacobian evaluations, matrix inversions and the total number of steps required to reach the end of the range of integration. We denote these by F_i, J_i, I_i and N_i (i = 1,2,3,4) for methods 1.-4. respectively. For all of our five test problems it was found that the number of steps taken by method 1 was always the largest of the four methods and the number of steps taken by method 3 was always the smallest. In view of this it would seem to be particularly appropriate when listing our results to give the ratios $F_1:F_3$, $J_1:J_3$, etc. and this is done. In **Tables 1-5** we give the details of the main computational effort involved with the four methods which we have considered and as can be seen method 1 performs rather poorly. Bearing in mind the L- stability of methods 3 and 4 it would seem to be advisable to use method 3 if function calls are very expensive compared with other operations and method 4 if Jacobian calls are the most expensive operations.

3.3. *Performance of the error estimation procedure*

It was found that both types of error estimate (i.e. the Merson type used by Haines and the $h - 2h$ type used by the other methods) were satisfactory for all problems run and in particular neither considerably underestimated the local truncation error (which is the dangerous case) at any stage. There is, however, always the nagging doubt that Merson type error estimates, which are strictly valid only for linear systems, may perform badly and produce disastrous results on certain classes of problems. A problem for which poor results are obtained has been described in the non-stiff case by England (1969) and in the stiff case problems may no doubt also be found for which the Merson error estimate performs badly.

3.4. *Conclusions*

In conclusion we may only repeat that in practice (certainly in the third order case and probably in general) it seems to be more efficient to derive schemes requiring as few function or Jacobian evaluations as possible and to use them with an $h - 2h$ error estimate rather than to derive one step methods containing 'built-in' error estimates.

References

ALLEN, R. H. and POTTLE, C. (1966). Stable integration methods for electronic circuit analysis with widely separated time constants, *Proc. Sixth Annual Allerton Conf. on Circuit and System Theory*, T. Trick and R. T. Chien, Eds, pp. 311-320.

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

CASH, J. R. (1976). Semi-implicit Runge-Kutta procedures with error estimates for the numerical integration of stiff systems of ordinary differential equations, *JACM*, Vol. 23, pp. 455-460.

ENGLAND, R. (1969). Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations, *The Computer Journal*, Vol. 12, pp. 166-170.

ENRIGHT, W. H., HULL, T. E. and LINDBERG, B. (1975). Comparing numerical methods for stiff systems of ODE's, *BIT*, Vol. 15, pp. 10-48.

HAINES, C. (1969). Implicit integration processes with error estimates for the numerical solution of differential equations, *The Computer Journal*, Vol. 12, pp. 183-187.

LAPIDUS, L. and SEINFELD, J. H. (1971). *Numerical Solution of Ordinary Differential Equations*, Academic Press.

LIEM, C. B. (1976). An Evaluation of Haines' Method for the Numerical Integration of Stiff Systems of Ordinary Differential Equations, M.Sc. Thesis, Dept. of Mathematics, Imperial College, University of London.

MERSON, R. H. (1957). An operational method for the study of integration processes, *Proc. Symp. on Data Processing*, Weapons Research Establishment, Salisbury, South Australia.

ROSENBROCK, H. H. (1963). General implicit processes for the numerical solution of differential equations, *The Computer Journal*, Vol. 5, pp. 329-330.