A Class of Implicit Runge-Kutta Methods for the Numerical Integration of Stiff Ordinary Differential Equations

J. R. CASH

Imperial College, London, England

ABSTRACT One-step methods similar in design to the well-known class of Runge-Kutta methods are developed for the efficient numerical integration of both stiff and nonstiff systems of first-order ordinary differential equations The algorithms developed combine accuracy in the limit $h \to 0$ with a large region of absolute stability and are demonstrated by direct application to certain particular examples.

KEY WORDS AND PHRASES: stiff ordinary differential equations, Runge-Kutta methods, Padé approximations, *A*-stability, Newton iteration, predictor-corrector schemes

CR CATEGORIES: 5 17

1. *Introduction.* This paper is concerned with the approximate numerical integration of systems of ordinary differential equations of the form

$$\dot{x} = f(t, x), \quad x(0) = x_0, \qquad t = nh. \tag{1.1}$$

In this section and in Section 2 the problem of developing algorithms suitable for the approximate numerical integration of systems of the form (1.1) with widely separated time constants, usually referred to as stiff [5], is considered; in Section 3 it is shown that these algorithms are also efficient for the integration of a large class of nonstiff systems. Locally the time constants associated with (1.1) are minus the reciprocals of the real parts of the eigenvalues of the Jacobian matrix

$$J = J(t) = f_x(t, x(t)), \tag{1.2}$$

where $x(t)$ is the particular solution of (1.1) which we wish to compute. Assuming that $J(t)$ is in some sense slowly varying, the solutions of eq. (1.1) are approximately exponentials of the form

$$x(t) = x(0) \exp(-\lambda t), \tag{1.3}$$

where λ is a simple eigenvalue of the Jacobian matrix $J(t)$. On a net of discrete points $t = nh$ the solution (1.3) satisfies the relation

$$x(t_{n+1}) = \exp(-q)x(t_n), \tag{1.4}$$

where $q = \lambda h$. It is desirable that formulas to be used for the numerical integration of stiff systems of ordinary differential equations should be stable for a large region of the complex plane as well as accurate in some neighborhood of the origin. Most explicit integration formulas have a bounded region of stability; in particular, it is well known

Author's address: Department of Mathematics, Imperial College, South Kensington, London S.W. 7, England.

that fourth-order explicit Runge-Kutta methods have a region of stability which intersects the imaginary and the negative real axis at $| h\lambda | \approx 2.7$ (see [10, p. 120]). This stability problem can be overcome by considering implicit Runge-Kutta formulas. In particular, Ehle [7] has shown that the R-stage implicit Runge-Kutta formulas developed by Butcher [2, 3] are all A-stable in the sense of Dahlquist [6]. Such methods do, however, suffer from a serious practical disadvantage in that the solution of the nonlinear implicit equations occurring at each time step is considerably harder to achieve than in the case of linear multistep methods. Intermediate between implicit and explicit Runge-Kutta methods are the semi-implicit Runge-Kutta methods developed by Rosenbrock [12] and further refined by Haines [8]. The motivation behind their approach is based on obtaining A-stable Runge-Kutta methods while still maintaining computational efficiency, i.e. avoiding the necessity to iterate. Semi-implicit A-stable Runge-Kutta methods do, however, seem to be much more difficult to derive than A-stable implicit Runge-Kutta methods, although Calahan [4] has developed a two-substitution method of order three and Allen and Pottle [1] have derived A-stable methods up to order four. In this paper we consider the derivation of L-stable formulas based on a class of methods similar in concept to the classical Runge-Kutta methods in that the solution at the point t_{n+1} is obtained from the solution at the point t_n using evaluations of the function $f(t, x)$ within the range (t_n, t_{n+1}). As a result it is only necessary to solve a system of algebraic equations with the single unknown x_{n+1} instead of the much larger system which has to be solved with most implicit Runge-Kutta methods. A method for the solution of these algebraic equations is considered in Section 3, and the algorithms are shown to compare favorably with certain existing methods for some particular systems in Section 4.

2. In this section we consider the approximate numerical integration of system (1.1) using a class of algorithms of the form

$$\begin{aligned} x_{n+1} - x_n &= h \sum_{p=1}^{m} C_p k_p\,, \\ k_i &= f\left(t_n + e_i h,\ x_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right),\ 1 \le i \le r, \text{ and} \\ k_i &= f\left(t_{n+1} + e_i h,\ x_{n+1} + h \sum_{j=r}^{i-1} a_{ij} k_j\right),\ r + 1 \le i \le m. \end{aligned} \tag{2.1}$$

These algorithms maintain computational efficiency, especially for high order equations, since each k_i is given explicitly either in terms of known quantities in the case where $1 \le i \le r$ or in terms of the single unknown x_{n+1} in the case $r + 1 \le i \le m$. As a result only one system of algebraic equations with x_{n+1} as the single unknown needs to be solved at each time step. It can easily be shown that two necessary conditions for the class of algorithms defined by (2.1) to yield A- or L-stable methods are, first, that none of the k_i for $1 \le i \le r$ are allowed in the expressions for k_i for $r + 1 \le i \le m$, and second, that r must be chosen so that $r \le [\frac{1}{2}m]$.

We confine our attention first of all to formulas of the form (2.1) with $r = 0$. Putting $m = 1$ we obtain the well-known backward Euler method which is L-stable and has order of accuracy $p = 1$. Putting $m = 2$, $r = 0$ in (2.1) we obtain an integration procedure of the form

$$x_{n+1} - x_n = h\{C_1 k_1 + C_2 k_2\}, \tag{2.2}$$

where

$$k_1 = f(t_{n+1}, x_{n+1}) \text{ and } k_2 = f(t_{n+1} + ah, x_{n+1} + ahk_1). \tag{2.3}$$

A second-order algorithm of the form (2.2) may be derived in the usual way by expanding both sides of eq. (2.2) in powers of h by a Taylor series about the point x_{n+1} and choosing the constants so that the coefficients of h^0, h, and h^2 are all zero. The identities derived are

$$1 = C_1 + C_2, \quad \tfrac{1}{2} = C_1 + (1+a)C_2, \tag{2.4}$$

the (infinitely many) solutions of which define a class of second-order algorithms of the form (2.1).

There is, however, an alternative derivation which demonstrates clearly the relation between formulas of the form (2.1) with $r = 0$ and certain classical explicit Runge-Kutta formulas. Equation (2.2) may be regarded as a "backward" version of the classical two-stage explicit Runge-Kutta formulas since it is equivalent to using a classical formula with step length $-h$. Consider, for example, the explicit Runge-Kutta formula

$$x_{n+1} - x_n = h\{\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2\}, \quad k_1 = f(t_n, x_n), \; k_2 = f(t_n + \tfrac{2}{3}h, x_n + \tfrac{2}{3}hk_1) \tag{2.5}$$

due to Heun. Replacing h 6g $-h$ in (2.5), we obtain the equation

$$x_n - x_{n-1} = h\{\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2\}, \quad k_1 = f(t_n, x_n), \; k_2 = f(t_n - \tfrac{2}{3}h, x_n - \tfrac{2}{3}hk_1), \tag{2.6}$$

which is a second-order algorithm of the form (2.1). We now examine the region of stability of scheme (2.6) and show how it can be determined immediately from that of the classical Runge-Kutta method (2.5). Applying algorithm (2.5) to the scalar test equation $\dot{x} = \lambda x$, where λ is a complex constant with negative real part, and putting $q = \lambda h$ we obtain an expression of the form

$$x_{n+1} = R(q)x_n, \tag{2.7}$$

where $R(q) = 1 + q + \frac{1}{2}q^2$ which is the (2, 0) Padé approximation to e^q. Similarly, applying algorithm (2.6) to the same scalar test equation we obtain a relation of the form

$$x_n = \bar{R}(q)x_{n-1}, \tag{2.8}$$

where, in order to determine $\bar{R}(q)$ from $R(q)$, we need only to reverse the sign of q in $R(q)$ and invert the fraction, i.e. $\bar{R}(q) = 1/(1 - q + \frac{1}{2}q^2)$, which is the (0, 2) Padé approximation to e^q. Hence it follows that the stability region of the backward formula (2.6) is the complementary set of the image of that of the classical formula (2.5) reflected in the imaginary axis. It now follows that, given any explicit Runge-Kutta formula, we may derive a backward formula of the form (2.1) with $r = 0$ by replacing h by $-h$ in the original equation, and the region of stability of the backward formula may be derived directly from that of the classical formula by taking the complementary set of its image in the imaginary axis. For example, the equation

$$\begin{aligned} x_{n+1} - x_n &= h\{\tfrac{1}{4}k_1 + \tfrac{3}{4}k_3\}, \\ k_1 &= f(t_{n+1}, x_{n+1}), \; k_2 = f(t_{n+1} - \tfrac{1}{3}h, x_{n+1} - \tfrac{1}{3}hk_1), \\ k_3 &= f(t_{n+1} - \tfrac{2}{3}h, x_{n+1} - \tfrac{2}{3}hk_2) \end{aligned} \tag{2.9}$$

is a backward version of Heun's third-order method, and the equation

$$\begin{aligned} x_{n+1} - x_n &= \tfrac{1}{6}h\{k_1 + 2k_2 + 2k_3 + k_4\}, \\ k_1 &= f(t_{n+1}, x_{n+1}), \; k_2 = f(t_{n+1} - \tfrac{1}{2}h, x_{n+1} - \tfrac{1}{2}hk_1), \\ k_3 &= f(t_{n+1} - \tfrac{1}{2}h, x_{n+1} - \tfrac{1}{2}hk_2), \; k_4 = f(t_{n+1} - h, x_{n+1} - hk_3) \end{aligned} \tag{2.10}$$

is a backward version of a well-known fourth-order Runge-Kutta method. Although these equations have very large regions of stability, including the negative real axis, they are not quite L-stable due to a very small region of instability near the imaginary axis. In order to derive an L-stable method we consider an equation of the form

$$\begin{aligned} x_{n+1} - x_n &= h\sum_{i=1}^{4} C_i k_i, \\ k_1 &= f(t_{n+1}, x_{n+1}), \; k_2 = f(t_{n+1} + ah, x_{n+1} + ahk_1), \\ k_3 &= f(t_{n+1} + bh, x_{n+1} + dhk_1 + (b-d)hk_2), \; k_4 = f(t_n, x_n), \end{aligned} \tag{2.11}$$

where a, b, d, and the c_i are constants to be determined. Scheme (2.11) is of the general form (2.1) with $m = 4$, $r = 1$. Expanding scheme (2.11) in a two-dimensional Taylor series about the point t_{n+1}, x_{n+1} and equating powers of h up to and including terms in h^3, we obtain the equations

$$\begin{aligned} 1 &= C_1 + C_2 + C_3 + C_4, \; -\tfrac{1}{2} = C_2a + C_3b - C_4, \\ \tfrac{1}{3} &= C_2a^2 + C_3b^2 + C_4, \; \tfrac{1}{6} = C_3(b - d)a + C_4/2, \end{aligned} \tag{2.12}$$

any solution of which yields a third-order algorithm of the form (2.1). Applying algorithm (2.11) to the scalar test equation $\dot{x} = \lambda x$, where λ is a complex constant with negative real part, we obtain the relation

$$x_{n+1} = \left\{\frac{1 + C_4q}{1 - (C_1 + C_2 + C_3)q - (C_2a + C_3b)q^2 - C_3(b - d)aq^3}\right\} x_n. \tag{2.13}$$

One particular algorithm of the form (2.11) satisfying relations (2.12) is the scheme

$$\begin{aligned} x_{n+1} - x_n &= h\{\tfrac{1}{4}k_2 + \tfrac{1}{2}k_3 + \tfrac{1}{4}k_4\}, \\ k_1 &= f(t_{n+1}, x_{n+1}), \; k_2 = f(t_{n+1} - \tfrac{1}{3}h, x_{n+1} - \tfrac{1}{3}hk_1), \\ k_3 &= f(t_{n+1} - \tfrac{1}{3}h, x_{n+1} - \tfrac{1}{12}hk_1 - \tfrac{1}{4}hk_2), \; k_4 = f(t_n, x_n). \end{aligned} \tag{2.14}$$

Using these values of the parameters we obtain a relationship of the form

$$x_{n+1} = R_T{}^S x_n, \text{ where } R_T{}^S = (1 + \tfrac{1}{4}q)/(1 - \tfrac{3}{4}q + \tfrac{1}{4}q^2 - \tfrac{1}{24}q^3). \tag{2.15}$$

Since $R_T{}^S$ is the (1, 3) Padé approximation to $\exp(q)$, we have from Ehle's results that algorithm (2.14) is L-stable. A second set of parameters satisfying relations (2.12) is

$$a = -0.5, \; b = -0.5, \; d = 0, \; C_1 = C_4 = \tfrac{1}{6}, \; C_2 = C_3 = \tfrac{1}{3},$$

yielding the algorithm

$$\begin{aligned} x_{n+1} - x_n &= \tfrac{1}{6}h\{k_1 + 2k_2 + 2k_3 + k_4\}, \\ k_1 &= f(t_{n+1}, x_{n+1}), \; k_2 = f(t_{n+1} - \tfrac{1}{2}h, x_{n+1} - \tfrac{1}{2}hk_1), \\ k_3 &= f(t_{n+1} - \tfrac{1}{2}h, x_{n+1} - \tfrac{1}{2}hk_2), \; k_4 = f(t_n, x_n). \end{aligned} \tag{2.16}$$

Both formulas (2.14) and (2.16) are virtually three-substitution third-order methods since at all points $t = nh$ apart from $n = 0$ the value of k_4 is given immediately as the value of k_1 at the previous step. From (2.13) it follows that eq. (2.16) applied to the scalar test equation $x = \lambda x$ yields the relation

$$\begin{aligned} x_{n+1} &= \{(1 + \tfrac{1}{6}q)/(1 - \tfrac{5}{6}q + \tfrac{1}{3}q^2 - \tfrac{1}{12}q^3)\}x_n, \; q = \lambda h \\ &= R(q)x_n \quad \text{say.} \end{aligned} \tag{2.17}$$

By definition eq. (2.16) is L-stable if and only if $|R(q)| < 1$ for all complex q with $\mathrm{Re}(q) < 0$. This in turn follows if and only if the following two conditions hold:

(i) $|R(q)| \leq 1$ on $\mathrm{Re}(q) = 0$,

(ii) $R(q)$ is analytic for $\mathrm{Re}(q) < 0$.

Condition (i) ensures that the rational function $R(q)$ is analytic at $q = -\infty$ and then conditions (i) and (ii) guarantee L-stability by the maximum modulus theorem. It can easily be verified that the approximation $R(q)$ in (2.17) satisfies both (i) and (ii), proving (2.16) to be L-stable.

It is clearly possible to develop higher order formulas of the form (2.1), but the author has made no attempt to do so in the present paper although it is a topic which seems to merit further research.

3. In this section we consider algorithms suitable for the numerical solution of the implicit (and in general nonlinear) algebraic equations arising at each time step from the use of algorithms of the form (2.1). In order to demonstrate the algorithms proposed we consider the solution of the implicit equations arising from the use of eq. (2.6), which may be written in the form (if $n + 1$ is substituted for n):

$$F(x_{n+1}) \equiv x_{n+1} - x_n - h\{\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2\} = 0. \tag{3.1}$$

If the system of equations (1.1) is not stiff in any particular region of the range of integration, the solution of (3.1) may be generated by means of the iteration scheme $x_{n+1}^{(p)} = x_n + hg(t_{n+1}, x_{n+1}^{(p-1)})$, where $g(t, x) = \tfrac{1}{4}f(t, x) + \tfrac{3}{4}f(t_{n+1} - \tfrac{2}{3}h, x_{n+1} - \tfrac{2}{3}hk_1)$. This iteration scheme will converge providing that h is chosen to satisfy the inequality $Lh < 1$, where L is the Lipschitz constant of $g(t, x)$ with respect to x. If the system (1.1) is stiff this inequality imposes an unacceptable restriction on the step size h, and in this case the single unknown vector x_{n+1} may be generated using the iterative scheme

$$J^{(p-1)}\{x_{n+1}^{(p)} - x_{n+1}^{(p-1)}\} + F(x_{n+1}^{(p-1)}) = 0, \tag{3.2}$$

where $J^{(p-1)}$ is a matrix if $\dot{x} = f(t, x)$ is a system. We do not specify $J^{(p-1)}$ at this stage stage except to impose the restriction that it should be nonsingular so that scheme (3.2) may be solved for $x_{n+1}^{(p)}$. We note that if we choose $J^{(p-1)} = (\partial F/\partial x)(t_{n+1}, x_{n+1}^{(p-1)})$, then scheme (3.2) reduces to the well-known Newton iteration scheme for the solution of a system of nonlinear equations. If a sequence of matrices $J^{(p-1)}$ can be chosen so that scheme (3.2) converges, then it does so to a solution of (3.1). Hence we may use an estimate for $\partial F/\partial x$ which need not be evaluated for each iteration or even for each time step if the Jacobian matrix $\partial F/\partial x$ changes slowly. The Jacobian matrix $\partial F/\partial x$ is given recursively in terms of the matrix $\partial f/\partial x$ evaluated at two distinct points.

Putting $A_1 = (\partial f/\partial x)\ (t_{n+1}, x_{n+1})$, $A_2 = (\partial f/\partial x)\ (t_{n+1} - \tfrac{2}{3}h, x_{n+1} - \tfrac{2}{3}hk_1)$, we have

$$\partial F/\partial x = I - \tfrac{1}{4}hA_1 - \tfrac{3}{4}hA_2 + \tfrac{1}{2}h^2A_2A_1. \tag{3.3}$$

So if algorithm (2.3) is used to integrate system (1.1), the Jacobian matrix $\partial f/\partial x$ needs to be evaluated at most twice, and one system of linear algebraic equations needs to be solved, for each iteration. Exactly how serious the need is to compute these extra function values depends, of course, on how costly the functions $f(t, x)$ and $(\partial f/\partial x)(t, x)$ are to evaluate, but it was found in practice that the algorithms developed in this paper are efficient compared with linear multistep methods except in the case where the time taken to evaluate the functions $f(t, x)$ and $(\partial f/\partial x)(t, x)$ is much larger than the time taken to obtain the solution of the system of linear algebraic equations defined by (3.2). Using the initial approximation $x_{n+1}^{(0)} = x_n$, the finally accepted value at the point t_n, it was found that the iteration scheme usually converged in three iterations. If scheme (3.2) failed to converge to the prescribed degree of precision in three iterations, the matrix $(\partial f/\partial x)(x_{n+1}^{(0)})$ was reevaluated and set equal to $J^{(0)}$. If the iteration scheme failed to converge a second time, the step was halved and the process was repeated. This scheme is necessarily finite since the scheme will always converge for sufficiently small h.

4. In this section we present some numerical results obtained using the algorithms derived in Section 2. It will be shown that the algorithms compare favorably with explicit Runge-Kutta methods and predictor-corrector methods for the two particular problems considered, due to the weak stability properties of the explicit methods. It was also found that the algorithms derived in Section 2 compare favorably with the fully implicit Runge-Kutta methods derived by Butcher for the same two problems. As previously mentioned, Ehle [7] has shown that Butcher's R-stage fully implicit Runge-Kutta methods of order $2R$ are all A-stable. Such methods do, however, suffer from the serious practical disadvantage that if an R-stage fully implicit Runge-Kutta method is applied to an m-dimensional system of stiff ordinary differential equations, then a system of mR nonlinear simultaneous algebraic equations will have to be solved at each time step by some form

of Newton iteration. Furthermore, these iterations will converge only if a suitable initial iterate can be found for $k_1, k_2, \cdots, k_R$ and this may not always be the case, especially if the functions k_i vary rapidly with x_{n+1}. The algorithms described in Section 2 do not suffer from this disadvantage since there is only one m-dimensional system of equations, with x_{n+1} as the unknown, to be solved at each time step, and the initial approximation $x_{n+1}^{(0)} = x_n$, the final accepted value at the point $t = nh$, usually provides a sufficiently accurate first approximation to x_{n+1} for scheme (3.2) to converge to the prescribed degree of precision in three iterations. We note that for stiff problems this initial choice $x_{n+1}^{(0)} = x_n$ for use in Newton's method with an h such that $Lh \gg 1$ makes sense only during the asymptotic phase (i.e. after the rapid transients have died out). During the initial phase when the transients have not died out it is necessary to use a step length h such that $Lh \leq 1$. For the above reasons, fully implicit Runge-Kutta methods have been found to be seldom competitive with other methods for integrating stiff systems of equations. If instead we consider semi-implicit Runge-Kutta methods, the mR equations split into R sets of m equations. It seems, however, to be more difficult to derive A-stable semi-implicit Runge-Kutta schemes than it is to derive A-stable fully implicit Runge-Kutta schemes. Semi-implicit Runge-Kutta schemes do, however, still suffer from the serious practical disadvantage that with an s-stage process the Jacobian matrix, in general, has to be evaluated and inverted at s distinct points for each step of the calculation, although Allen and Pottle [1] have derived some A-stable s-stage formulas which require less than s Jacobian evaluations per time step. Since it is not normally known at the outset whether a given system is stiff or not, it is important to develop algorithms which are efficient for both stiff and nonstiff problems when developing a general purpose algorithm. Semi-implicit Runge-Kutta methods do not satisfy this condition, since it is clear that they are inefficient compared with the algorithms derived in Section 2 and with linear multistep methods for the numerical integration of nonstiff systems. Associated with an s-stage semi-implicit Runge-Kutta method there are still s distinct sets of m simultaneous algebraic equations to be solved, whereas associated with the algorithms derived in Section 2 there is only one set of m simultaneous algebraic equations to be solved and this may be done very efficiently using the predictor-corrector technique described in Section 3. It seems that when using semi-implicit Runge-Kutta schemes it is necessary to incorporate them into a package which has the option of using an alternative scheme if it is detected that the system is not stiff.

Table I gives the results obtained for the solution of the system

$$\begin{aligned}
\dot{x}_1 &= -10^4 x_1 + x_2^4 - 2x_3^2 + x_4^2 - x_5, \\
\dot{x}_2 &= -\tfrac{1}{2}x_2 + x_1 - x_3^2, \\
\dot{x}_3 &= -0.01x_2^2, \\
\dot{x}_4 &= -x_3 + x_1^3 - x_5^3, \\
\dot{x}_5 &= -x_1 - x_3 x_4,
\end{aligned} \tag{4.1}$$

where $x_1(0) = x_3(0) = x_4(0) = x_5(0) = 1$, $x_2(0) = 10$. The solution of this system was required at the point $t = 1$ with an absolute accuracy of 10^{-8}. This system is stiff throughout the range of interest and has a stiffness ratio 10^4 at the point $t = 0$.

TABLE I

$\dfrac{\text{Computation time for trapezoidal rule}}{\text{Computation time for algorithm (2 14)}}$	= 1.4
$\dfrac{\text{Computation time for Adams method}}{\text{Computation time for algorithm (2 14)}}$	= 12.4
$\dfrac{\text{Computation time for Runge-Kutta method}}{\text{Computation time for algorithm (2.14)}}$	= 11 8

The algorithms tested were the explicit fourth-order Runge-Kutta scheme

$$\begin{aligned} x_{n+1} - x_n &= \tfrac{1}{6}h\{k_1 + 2k_2 + 2k_3 + k_4\}, \\ k_1 &= f(t_n, x_n),\ k_2 = f(t_n + \tfrac{1}{2}h, x_n + \tfrac{1}{2}hk_1), \\ k_3 &= f(t_n + \tfrac{1}{2}h, x_n + \tfrac{1}{2}hk_2),\ k_4 = f(t_{n+1}, x_n + hk_3); \end{aligned} \tag{4.2}$$

the fourth-order Adams predictor-corrector method

$$\begin{aligned} \bar{x}_{n+1} &= x_n + \tfrac{1}{24}h\{55f(t_n, x_n) - 59f(t_{n-1}, x_{n-1}) + 37f(t_{n-2}, x_{n-2}) - 9f(t_{n-3}, x_{n-3})\}, \\ x_{n+1} &= x_n + \tfrac{1}{24}h\{9f(t_{n+1}, \bar{x}_{n+1}) + 19f(t_n, x_n) - 5f(t_{n-1}, x_{n-1}) + f(t_{n-2}, x_{n-2})\}, \end{aligned}$$

using one corrector evaluation and using the fourth-order Runge-Kutta method (4.2) to start; the trapezoidal rule; and algorithm (2.14). The ratios of the times taken for these four methods are given in Table I. As can be seen, algorithm (2.14) compares favorably with the other three algorithms tested for this particular problem.

As a second example we consider the numerical integration of the system

$$x = -10004x + 10000y^4, \quad \dot{y} = -y + x - y^4,$$

with initial conditions $x(0) = y(0) = 1$ and using a step length $h = 0.125$. The algorithms compared were (2.6), (2.14), the trapezoidal rule, and Calahan's method [4]. As can be seen from Table II the solution for $x(t)$ is represented by an oscillatory solution when using the trapezoidal rule and Calahan's method due to the incorrect asymptotic root behavior of these two methods. In particular, the trapezoidal rule applied to the scalar test equation $\dot{x} = \lambda x$, with λ a complex constant with negative real part, yields the relation

$$\begin{aligned} | x_{n+1}/x_n | &= | (1 + \tfrac{1}{2}h\lambda)/(1 - \tfrac{1}{2}h\lambda) | \\ &= \{[1 + h\mathrm{Re}(\lambda) + \tfrac{1}{4}h^2 | \lambda^2 |]/[1 - h\mathrm{Re}(\lambda) + \tfrac{1}{4}h^2 | \lambda^2 |]\} \to 1 \text{ as } h\lambda \to -\infty. \end{aligned}$$

We note, however, that in the case of the trapezoidal rule this problem can be overcome by the use of extrapolation and smoothing [11]. In general, however, it seems that A-stable methods which are not damped as $h\lambda \to -\infty$ (i.e. are not L-stable, see [9, p. 236]) will, in many cases, be unsatisfactory when dealing with excessively stiff systems. For L-stable algorithms, such as (2.6) and (2.14), this problem is overcome since by definition we have $| x_{n+1}/x_n | \to 0$ as $h\lambda \to -\infty$, which agrees with the true asymptotic behavior of the solution. For these reasons, when developing general purpose algorithms, it seems desirable in many cases to develop L-stable rather than nonasymptotically damped A-stable schemes for the solution of inherently stable schemes. The numerical

TABLE II

t		Trapezoidal rule	Method (2 6)	Method (2 14)	Calahan's method	True solution
0 625	$e(x)$	27626	−49311	−198	882900	0 08208500
	$e(y)$	43763	−79500	20	5086	0 53526143
1 250	$e(x)$	3655	−8080	−15	109403	0 00673795
	$e(y)$	46676	−85131	20	6046	0.28650480
1.875	$e(x)$	1267	−997	−2	28755	0 00055308
	$e(y)$	37522	−68389	15	4440	0 15335497
2.500	$e(x)$	−630	−109	0	−5448	0 00004540
	$e(y)$	26761	−48840	9	3170	0 08208499
3 125	$e(x)$	712	−11	0	1170	0 00000373
	$e(y)$	17894	−32700	7	1120	0 04393693
3 750	$e(x)$	−693	−1	0	−238	0 00000031
	$e(y)$	11489	−21018	5	1362	0.02351775
4 375	$e(x)$	683	0	0	50	0.00000003
	$e(y)$	7171	−13135	3	850	0.01258814
5.000	$e(x)$	−672	0	0	−10	0.00000000
	$e(y)$	4385	−8040	2	520	0 00673795

results obtained for the solution of system (4.3) are given in Table II. The quantity $e(x)$ given for any value of t defines the error, multiplied by 10^8, in the value of x for that value of t. Similarly, $e(y)$ gives the error, multiplied by 10^8, in the value of y obtained.

5. *Conclusion.* The purpose of this paper was to develop efficient numerical algorithms similar in design to the classical Runge-Kutta methods suitable for the integration of both stiff and nonstiff systems of inherently stable ordinary differential equations. The algorithms developed in Section 2 combine the desirable property of L-stability with a reasonable order of accuracy while still maintaining computational efficiency and preserving the one-step nature of the algorithms. The efficiency of the algorithms developed compared with certain other algorithms for some particular problems is demonstrated by the results given in Section 4.

REFERENCES

1 ALLEN, R. H , AND POTTLE, C 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 , 1966, pp 311–320.

2 BUTCHER, J. C Coefficients for the study of Runge-Kutta integration processes. *J. Australian Math Soc. 3* (1963), 202–206

3. BUTCHER, J C Implicit Runge-Kutta processes *Math Comput 18* (1964), 50–64

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

5 CURTIS, C F , AND HIRSCHFELDER, J. O Integration of stiff equations. *Proc. Nat. Acad. Sci U.S A 38* (1952), 235–243

6. DAHLQUIST, G G A special stability problem for linear multistep methods *BIT 3* (1963), 27–43

7. EHLE, B L. On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems. Res Rep CSRR 2010, Dep. of Applied Analysis and Computer Science, U of Waterloo, Waterloo, Ont , Canada, 1969.

8. HAINES, C F Implicit integration processes with error estimates for the numerical solution of differential equations *Computer J 12* (1968), 183–187

9. LAMBERT, J D *Computational Methods in Ordinary Differential Equations.* Wiley, New York, 1973.

10. LAPIDUS, L., AND SEINFELD, J. H. *Numerical Solution of Ordinary Differential Equations.* Academic Press, New York, 1971

11 LINDBERG, B. On smoothing and extrapolation for the trapezoidal rule. Rep. Royal Inst. Technol , Stockholm, Sweden, Aug 1969

12. ROSENBROCK, H H. Some general implicit processes for the numerical solution of differential equations. *Computer J 5* (1963), 329–330

RECEIVED AUGUST 1973; REVISED JANUARY 1975