Journal Title: IMA Journal of Numerical Analysis

Volume: 3
Issue: 3
Month/Year: 1983
Pages: 273-289

Article Author: Cameron

Article Title: Solution of differential-algebraic systems using diagonally implicit runge-kutta methods

IMA Journal of Numerical Analysis (1983) **3**, 273–289

Solution of Differential–Algebraic Systems Using Diagonally Implicit Runge–Kutta Methods

IAN T. CAMERON

Department of Chemical Engineering and Chemical Technology, Imperial College of Science and Technology, London SW7

[Received 7 July 1982 and in revised form 2 November 1982]

Diagonally Implicit Runge–Kutta (DIRK) methods are developed and applied to differential–algebraic systems arising from dynamic process simulation. In particular, an embedded family of DIRK methods is developed for implementation as a variable-step variable-order algorithm. The methods developed allow easy assessment of local solution error as well as the ability to change the order of approximation. The stability properties of the methods are chosen to make them suitable for use on stiff systems.

Some important aspects of implementation of DIRK methods are discussed within the context of the solution of differential–algebraic systems. The performance of this algorithm is compared with an alternative variable-order approach based on "triples" which allows the patching together of several fixed-order formulae. The results indicate that the fully embedded DIRK algorithm is generally more efficient than the algorithm based on "triples". Areas of further investigation in the context of differential–algebraic systems are outlined.

1. Introduction

THE DYNAMIC MODELLING of lumped parameter systems normally results in a set of ordinary differential equations coupled to a set of algebraic equations and subroutines of procedures. These procedures can be thermodynamic property routines or control algorithms, the presence of which adds significantly to the complexity of the equations. In the context of chemical engineering applications these differential–algebraic systems are generally non-linear. As well, other characteristics often displayed by these systems include stiffness, oscillatory behaviour and the presence of discontinuities. The discontinuities can be of two distinct types: time events and state events. Time events occur at prescribed values of the independent variable whilst the time when a dependent (or state) variable reaches a specified threshold defines a state event.

These characteristics put particular demands on the numerical procedures employed to generate the solution trajectories. No single numerical procedure is optimal for all circumstances, however reliability and robustness are necessary qualities of any method used to generate solutions for these types of problems.

The differential–algebraic system considered in this paper can be written as:

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(\mathbf{y}, \mathbf{z}, t), \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z}, t), \end{aligned} \tag{1.1}$$

0272–4979/83/030273 + 17 $03.00/0

where

$$\mathbf{y} \in R^m, \qquad \mathbf{f}: R^m \times R^n \times R \to R^m, \qquad t \in [a, b],$$
$$\mathbf{z} \in R^n, \qquad \mathbf{g}: R^m \times R^n \times R \to R^n,$$

and we denote the vector $\mathbf{y}$ as the differential variables and the vector $\mathbf{z}$ as the algebraic variables.

In general, the initial conditions for the differential variables will be given as $\mathbf{y}(a) = \mathbf{y}_0$ and it would then be necessary to establish the values of the algebraic variables at $t = a$. General conditions can be applied to ensure a unique solution of Equation (1.1) by assuming that the functions $\mathbf{f}$ and $\mathbf{g}$ satisfy Lipschitz conditions. That is, L_1 and L_2 exist such that

$$\begin{aligned} \|\mathbf{f}(\mathbf{y}_1, \mathbf{z}, t) - \mathbf{f}(\mathbf{y}_2, \mathbf{z}, t)\| &\leqslant L_1 \|\mathbf{y}_1 - \mathbf{y}_2\| \\ \|\mathbf{g}(\mathbf{y}, \mathbf{z}_1, t) - \mathbf{g}(\mathbf{y}, \mathbf{z}_2, t)\| &\leqslant L_2 \|\mathbf{z}_1 - \mathbf{z}_2\| \end{aligned} \tag{1.2}$$

for all $t \in [a, b]$ and all components in $\mathbf{y}$ and $\mathbf{z}$. These conditions are minimal and also require that the Jacobian of the algebraic system $(\partial g_i / \partial z_j)$ be of full rank.

This paper is concerned with the development and application of Diagonally Implicit Runge–Kutta (DIRK) methods to mixed systems like (1.1). The general s-stage DIRK method applied to the differential part of the equation system (1.1) has the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{f}(\mathbf{y}_{n,i}, \mathbf{z}_{n,i}, t_{n,i}), \tag{1.3a}$$

where

$$\mathbf{y}_{n,i} = \mathbf{y}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{f}(\mathbf{y}_{n,j}, \mathbf{z}_{n,j}, t_{n,j}) + \gamma h \mathbf{f}(\mathbf{y}_{n,i}, \mathbf{z}_{n,i}, t_{n,i}),$$
$$i = 1(1)s, \tag{1.3b}$$

with $\gamma = a_{ii}$; $i = 1(1)s$ and $t_{n,i} = t_n + c_i h$. The parameters of the method can be expressed in matrix form as

$$\begin{array}{c|ccccc} c_1 & a_{11} & & & & \\ \vdots & \vdots & & & & \\ c_s & a_{s1} & \cdots & & & a_{ss} \\ \hline & b_1 & \cdots & & & b_s \end{array} \tag{1.4}$$

where

$$c_i = \sum_j a_{ij}.$$

Several workers have studied this class of Runge–Kutta methods, most notably Nørsett (1974*a*), although Crouzeix (1975), Alexander (1977), Cooper & Sayfy (1979) and Cash (1979) have investigated various aspects of the methods. The numerical tests performed on these methods have illustrated their usefulness and efficiency on several difficult problems. The DIRK methods also possess a valuable range of stability properties which makes them potentially useful procedures for solving chemical engineering based problems. The methods are especially attractive for the simulation of start-up and shut-down studies of processes where many time events

are scheduled resulting in an interruption to the integration routine at each event. In these simulation problems, Runge–Kutta methods can maintain and recommence at high order whereas linear multistep methods are usually required to revert to first order for restart.

Almost without exception the DIRK methods presented in the literature have been fixed-order formulae. The possibility of formulating variable-step variable-order (VSVO) methods has not received much attention. In Section 2 several approaches to VSVO methods are discussed. As well, a family of DIRK methods is developed which can be easily used in VSVO mode. Section 3 discusses the application of DIRK methods to the solution of differential–algebraic systems whilst Section 4 outlines some important aspects of implementation. Finally, Section 5 gives details of some numerical experiments on several mixed systems.

2. Variable-step Variable-order Methods (VSVO)

The purpose of steplength and order variation is to minimize the over-all computational work required for a particular problem. Experiments using fixed-order Runge–Kutta methods indicate that high-order methods are more efficient at high accuracy because of truncation-error considerations, whilst low-order methods are generally more efficient at low accuracy. Hence it can be useful to vary both steplength and order to achieve optimal performance.

Recently, Cash & Liem (1980) presented a VSVO DIRK algorithm based on the concept of "triples". This essentially allows the patching together of several different fixed-order DIRK methods. Each "triple" consists of an integration formula $F^{(p)}$ (of order p), an embedded formula $F^{(p-1)}$ and an enveloping formula $F^{(p+1)}$. The embedded formula provides error estimation for the integration formula whilst the enveloping formula gives an error estimate for the higher-order method and thus an assessment of whether order change is worthwhile. If order change is deemed useful, a different "triple" is then used to continue the integration. However, error estimates for the same-order formula in different "triples" will be different and this can cause difficulties in the order-changing mechanism.

The results quoted by Cash & Liem on several small problems indicated the usefulness of the implementation over a fixed fourth-order code.

2.1 *An Embedded DIRK Family*

An alternative approach to the formulation of a VSVO code is to obtain a set of embedded formulae, so that by the successive addition of stages, higher-order approximations can be obtained. In the matrix notation of (1.4) we would have a sequence of methods as follows.

$$\begin{array}{c|c} c_1 & \gamma \\ \hline & b_1 \end{array} \qquad \begin{array}{c|cc} c_1 & \gamma & \\ c_2 & a_{21} & \gamma \\ \hline & d_1 & d_2 \end{array} \qquad \begin{array}{c|ccc} c_1 & \gamma & & \\ c_2 & a_{21} & \gamma & \\ c_3 & a_{31} & a_{32} & \gamma \\ \hline & e_1 & e_2 & e_3 \end{array} \qquad \begin{array}{c|cccc} c_1 & \gamma & & & \\ c_2 & a_{21} & \gamma & & \\ c_3 & a_{31} & a_{32} & \gamma & \\ c_4 & a_{41} & a_{42} & a_{43} & \gamma \\ \hline & f_1 & f_2 & f_3 & f_4 \end{array} \qquad \ldots (2.1)$$

Using these embedded methods an error estimate for the pth-order solution at t_{n+1} can be obtained by

$$\mathbf{E}_{n+1} = \|\mathbf{y}_{n+1}^{(p+1)} - \mathbf{y}_{n+1}^{(p)}\|,$$

where $\mathbf{y}_{n+1}^{(p)}$ and $\mathbf{y}_{n+1}^{(p+1)}$ are the numerical solutions of order p and $p+1$, respectively. If local extrapolation is used the solution proceeds with the higher-order solution $\mathbf{y}_{n+1}^{(p+1)}$. If an order-p method is being used then errors for orders $p, p-1, \ldots$ can be obtained from the embedded formulae whilst the next-highest order is obtained by computing the next-stage calculation. This allows easy assessment of the benefits of order changing based on local solution error. Since the diagonal element γ is identical for each order method the same iteration matrix $[\mathbf{I}-h\gamma\mathbf{J}]$ can be used in the Newton-like solution of the stage values $\mathbf{y}_{n,i}$.

Consideration is now given to the solution of such embedded systems given by (2.1).

2.2 *Order Conditions*

For simplicity the embedded methods are only considered up to four stages. Higher-order methods could be considered although they are much more complicated to construct. It is required that the order of each method be equal to the number of stages. The order conditions which must be solved can be derived using the Butcher series approach (see Hairer & Wanner, 1974; Nørsett, 1974*a*). In particular, for the sequence of methods given by (2.1) the following conditions apply.

Order 4:

$$\begin{aligned}
\textstyle\sum f_i &= 1 = p_1(\gamma)\\
\textstyle\sum f_i\tilde{c}_i &= \tfrac{1}{2}-\gamma = p_2(\gamma)\\
\textstyle\sum f_i\tilde{c}_i^2 &= \tfrac{1}{3}-\gamma+\gamma^2 = p_3(\gamma)\\
\textstyle\sum f_i a_{ij}\tilde{c}_j &= \tfrac{1}{6}-\gamma+\gamma^2 = p_4(\gamma)\\
\textstyle\sum f_i\tilde{c}_i^3 &= \tfrac{1}{4}-\gamma+3\gamma^2/2-\gamma^3 = p_5(\gamma)\\
\textstyle\sum f_i\tilde{c}_i a_{ij}\tilde{c}_j &= \tfrac{1}{8}-5\gamma/6+3\gamma^2/2-\gamma^3 = p_6(\gamma)\\
\textstyle\sum f_i a_{ij}\tilde{c}_j^2 &= \tfrac{1}{12}-2\gamma/3+3\gamma^2/2-\gamma^3 = p_7(\gamma)\\
\textstyle\sum f_i a_{ij}a_{jk}\tilde{c}_k &= \tfrac{1}{24}-\gamma/2+3\gamma^2/2-\gamma^3 = p_8(\gamma),
\end{aligned}$$

Order 3:

$$\begin{aligned}
\textstyle\sum e_i &= p_1(\gamma)\\
\textstyle\sum e_i\tilde{c}_i &= p_2(\gamma)\\
\textstyle\sum e_i\tilde{c}_i^2 &= p_3(\gamma)\\
\textstyle\sum e_i a_{ij}\tilde{c}_j &= p_4(\gamma),
\end{aligned} \qquad (2.2)$$

Order 2:

$$\begin{aligned}
\textstyle\sum d_i &= p_1(\gamma)\\
\textstyle\sum d_i\tilde{c}_i &= p_2(\gamma),
\end{aligned}$$

Order 1:

$$\sum b_i = p_1(\gamma),$$

where the summations are over $i, j, k = 1(1)s$ and with $\tilde{c}_i = c_i - \gamma$, being the row sums of the explicit part of the Runge–Kutta methods.

Many solutions exist for the parameter values in the underdetermined system of equations given by (2.2) However, it is preferable to impose certain constraints on the parameters to achieve the desired stability properties of each method as well as the values of the abscissae c_i, $i = 1(1)s$.

2.3 *Stability*

THEOREM 2.1 *The rational approximation to the exponential function for a DIRK method of order s is given by*

$$R(z) = \sum_{k=0}^{s} L_k^{(s-k)}\left(\frac{1}{\gamma}\right)(-\gamma z)^k/(1-\gamma z)^s, \tag{2.3}$$

where the Laguerre polynomial is defined as in Beyer (1978).

This rational approximation is a well-known result and was derived and studied extensively by Nørsett (1974*b*, 1975) and also by Wolfbrandt (1977). From (2.3) it is possible to establish the ranges of γ which produce A-acceptable rational approximations for DIRK methods of various orders. These are given in Table 1 up to order 5 (being also quoted by Burrage, 1978).

A choice of $\gamma \in [0{\cdot}5, 1{\cdot}06858]$ will give methods which are A-stable for order $p = s$, $s = 1(1)4$. Choosing $\gamma \in [0{\cdot}394339, 0{\cdot}5]$ yields A-stable methods for $p = s$, $s = 2(1)4$ with $p = 1$ having a small stability region in the left-hand plane. The particular choice of γ as a root of $L_3^{(0)}(1/\gamma) = 0$ in the interval $[0{\cdot}394339, 0{\cdot}5]$ makes $p = 3$ an L-stable method. The choice of γ in the lower interval also reduces the error constants in the local truncation error expressions for the methods $p = s$, $s = 2(1)4$.

It is important to note that the unstable regions of these DIRK methods are closed regions in the right-hand $h\lambda$-plane. Thus a large amount of the right-hand plane is stable. When applied to truly unstable problems these methods are unreliable.

It is possible to enlarge the unstable regions in the right-hand $h\lambda$-plane by moving γ closer to the boundary of the stability intervals given in Table 1. However, the value of $|R(z)|$ in the majority of the right-hand $h\lambda$-plane is generally close to unity. If the problem becomes unstable when some of the system eigenvalues move into the

TABLE 1

Order	γ interval
1	$[0{\cdot}5, \infty)$
2	$[0{\cdot}25, \infty)$
3	$[\frac{1}{3}, 1{\cdot}06858]$
4	$[0{\cdot}394339, 1{\cdot}28057]$
5	$[0{\cdot}246506, 0{\cdot}361801]$, $[0{\cdot}420785, 0{\cdot}473280]$

right-hand plane, the growth in the solution can be so slow that the instability is never detected. This is a serious deficiency with all implicit methods which have large stable regions in the right-hand plane.

2.4 *A Particular Solution*

Let us fix γ as a root of $L_3^{(0)}(1/\gamma) = 0$. It is also convenient to restrict the c_i values such that

$$c_i \in (0, 1), \quad i = 1(1)s, \tag{2.4}$$

so that derivative information is not being obtained beyond or behind the current steplength. This is especially important when discontinuities are present, since information beyond the steplength can be invalid. As well as condition (2.4) it is also convenient to locate one of the c_i values at the end of the interval and to do this it is possible to set $c_3 = 1{\cdot}0$. The location of c_3 at the end of the interval would aid the effective location of discontinuities in the dependent variables.

With these constraints, the following parameter values are obtained for the embedded methods given by (2.1):

$$\begin{aligned}
\gamma &= 0{\cdot}435866521508, & f_1 &= 0{\cdot}238148535874,\\
a_{21} &= -0{\cdot}403494298165, & f_2 &= 0{\cdot}190784762258,\\
a_{31} &= -0{\cdot}381596758045, & f_3 &= 0{\cdot}155701460900,\\
a_{32} &= 0{\cdot}945730236526, & f_4 &= 0{\cdot}415365240968,\\
a_{41} &= 0{\cdot}401916934763, & e_1 &= 0{\cdot}661090792671,\\
a_{42} &= -0{\cdot}110263523009, & e_2 &= 0{\cdot}131307259462,\\
a_{43} &= -0{\cdot}163386454770, & e_3 &= 0{\cdot}207601947867,\\
c_2 &= 0{\cdot}032372223343, & d_1 &= 1{\cdot}158945191501,\\
c_3 &= 1{\cdot}0, & d_2 &= -0{\cdot}158945191501,\\
c_4 &= 0{\cdot}564133478492, & b_1 &= 1{\cdot}0.
\end{aligned}$$

It is important to note that if condition (2.4) is relaxed and information beyond the current step is allowed, then making $c_3 = 1 + c_2$ saves a stage calculation if the current steplength has not changed from the previous step. Also, the value of nearly all the above parameters lies in the range $(-1, +1)$ which helps in reducing the effect of round-off errors as well as minimizing error propagation within the stage calculations.

3. Application to Differential–Algebraic Systems

To compute the stage calculations $\mathbf{y}_{n,i}$, $i = 1(1)s$, a modified Newton iteration is used. First, an "outer" iteration can be adopted for the differential system coupled to an "inner" iteration on the algebraic equations. Secondly, a "combined" iteration scheme can be established which iterates on both differential and algebraic variables simultaneously. Numerical experience with the first scheme is disappointing with frequent convergence failures. However, the combined iteration scheme has proved very reliable with no significant convergence problems.

The stage calculations $\mathbf{y}_{n,i}$ (1.3) can be written in the generic form

$$\mathbf{y} = h\gamma\mathbf{f}(\mathbf{y}, \mathbf{z}, t) + \boldsymbol{\psi}, \tag{3.1}$$

where $\boldsymbol{\psi}$ contains known (past) solution information. Applying a Newton iteration to the mixed system gives

$$\begin{bmatrix} \mathbf{I} - h\gamma\mathbf{J}_1 & -h\gamma\mathbf{J}_2 \\ -h\gamma\mathbf{G}_1 & -h\gamma\mathbf{G}_2 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}^{k+1} \\ \Delta\mathbf{z}^{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\psi} + h\gamma\mathbf{f}(\mathbf{y}^k, \mathbf{z}^k, t) - \mathbf{y}^k \\ h\gamma\mathbf{g}(\mathbf{y}^k, \mathbf{z}^k, t) \end{bmatrix}, \tag{3.2}$$

with

$$\Delta\mathbf{y}^{k+1} = \mathbf{y}^{k+1} - \mathbf{y}^k, \qquad \Delta\mathbf{z}^{k+1} = \mathbf{z}^{k+1} - \mathbf{z}^k,$$

and

$$\mathbf{J}_1 = (\partial f_i/\partial y_j), \qquad \mathbf{J}_2 = (\partial f_i/\hbar z_j),$$
$$\mathbf{G}_1 = (\partial g_i/\partial y_j), \qquad \mathbf{G}_2 = (\partial g_i/\partial z_j).$$

Using the iterative scheme (3.2) it is possible to solve for each stage calculation in turn. The final values of the differential variables $\mathbf{y}_{n+1}$ are obtained by the normal linear combination of the solution values as given by (1.3a). However, to compute $\mathbf{z}_{n+1}$ exactly it is necessary after having obtained $\mathbf{y}_{n+1}$ to iterate the subsystem of equations related to the algebraic system,

$$\frac{\partial \mathbf{g}}{\partial \mathbf{z}} \cdot \Delta\mathbf{z}^{k+1} = -\mathbf{g}(\mathbf{y}_{n+1}, \mathbf{z}^k_{n+1}, t_{n+1}). \tag{3.3}$$

This normally requires one or two iterations to achieve convergence since good starting values for $\mathbf{z}_{n+1}$ can be generated from previous solution values.

This extra iterative procedure (3.3) can be avoided by using DIRK methods with special structure. The basic DIRK method (1.3a, b) can be rewritten as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{b}^t\mathbf{A}^{-1}(\mathbf{Y}_{n,i} - \mathbf{Y}_n),$$

where

$$\mathbf{Y}_{n,i} = (\mathbf{y}_{n,1}, \ldots, \mathbf{y}_{n,s})^t$$

and

$$\mathbf{Y}_n = (\mathbf{y}_n, \ldots, \mathbf{y}_n)^t.$$

If the condition is imposed that $\mathbf{b}^t\mathbf{A}^{-1} = (0, 0, \ldots, 0, 1)$ then $\mathbf{y}_{n+1} = \mathbf{y}_{n,s}$, so that the last stage calculation becomes the final solution at the end of the current step. This also means that the accompanying algebraic variables are solved without further computation. In terms of the matrix of Runge–Kutta coefficients the above condition gives

$$b_i = a_{si}, \quad i = 1(1)s,$$

and hence $c_s = 1$.

If $\gamma = a_{ii}$, $i = 1(1)s$ is chosen appropriately, the resulting DIRK methods possess the property of strong S-stability as defined by Prothero & Robinson (1974) and developed by Alexander (1977). This more restrictive stability property gave superior performance over A-stable methods for a class of problems tested by Alexander. The strongly S-stable methods also form the basis of the VSVO DIRK algorithm of Cash & Liem (1980). In these methods the order is given by $p = s$ for $s = 1(1)3$ with five stages being required for a method of order 4.

4. Some Implementation Aspects of DIRK Codes for Differential–Algebraic Systems

The transcription of the basic formulae into a robust and reliable code is not a trivial matter. In this section some of the more important implementation aspects are briefly discussed.

4.1 *Solution of the Iterative Scheme*

For each stage calculation and iteration k a residual is calculated as

$$R^k = \left[\sum_{i=1}^{m} \left(\frac{\Delta y(i)^k}{W_{y(i)}} \right)^2 + \sum_{j=1}^{n} \left(\frac{\Delta z(j)^k}{W_{z(j)}} \right)^2 \right]^{\frac{1}{2}},$$

this being compared with a convergence tolerance ε. The quantities $W_{y(i)}$ and $W_{z(j)}$ are error weights for each component of $\mathbf{y}$ and $\mathbf{z}$, depending on whether the error is absolute, relative, mixed or related to the largest value of the component realized up to the current integration time.

We attempt to relate the convergence tolerance ε to the user-set local error tolerance τ by considering the propagation of error in the stage calculations and its effect on the computed solution and local error estimate. There are also further implications with respect to errors in the calculated steplength but these will not be investigated here. The accumulated stage errors are carried through into higher stages via the ψ term in (3.1). The propagation of errors is non-trivial and is related to the matrix of parameters (1.4) being used. From (3.1) the error in the scaled derivative for the ith stage calculation can be written as:

$$h\Delta\mathbf{f}(\mathbf{y}_i) = (\Delta\mathbf{y}_i - \Delta\boldsymbol{\psi}_i + \boldsymbol{\xi}_i)/\gamma, \tag{4.1}$$

where Δ represents the error between the true and computed values for the quantities $\mathbf{f}$, $\mathbf{y}_i$ and $\boldsymbol{\psi}_i$ with $\boldsymbol{\xi}_i$ being the residual. Now the accumulated error $\Delta\boldsymbol{\psi}_i$ in the stage is given by

$$\Delta\boldsymbol{\psi}_1 = 0 \quad \text{and} \quad \Delta\boldsymbol{\psi}_i = \sum_{j=1}^{i-1} a_{ij}(h\Delta(\mathbf{f}y_j)), \quad i > 1,$$

and this can be written after some tedious manipulation in terms of the method parameters, stage residual and stage error so that

$$\Delta\boldsymbol{\psi}_i = \sum_{j=1}^{i-1} c_{ij}(\Delta\mathbf{y}_j + \boldsymbol{\xi}_j), \quad i > 1,$$

with (4.2)

$$c_{ij} = \sum_{m=1}^{(i-j)} \frac{(-1)^m}{\gamma^m} \Big(\sum_{t \in T_m} a(t) \Big),$$

where a: $T_m \to R$ and T_m is set of all possible products of m parameters $a_{l_1 l_2} \ldots a_{l_{2m-1} l_{2m}}$ such that for each product, $l_1 = i$, $l_{2m} = j$ and $i > l_2 \geqslant l_3 \ldots \geqslant l_{2m-1} > j$.

A bound can now be established on the error in the scaled derivative (4.1) as

$$\|h\Delta\mathbf{f}(\mathbf{y}_i)\| \leqslant \left| \frac{1}{\gamma} \right| \cdot \left[\|\Delta\mathbf{y}_i\| + \|\Delta\boldsymbol{\psi}_i\| + \|\boldsymbol{\xi}_i\| \right], \tag{4.3}$$

with

$$||\Delta \mathbf{y}_i|| \leqslant \varepsilon$$

and from (4.2)

$$||\Delta \boldsymbol{\psi}_i|| \leqslant \sum_{j=1}^{i-1} |c_{ij}| \cdot [||\Delta \mathbf{y}_j|| + ||\boldsymbol{\xi}_j||]. \tag{4.4}$$

It can be established that $||\boldsymbol{\xi}_i|| \simeq ||\Delta \mathbf{y}_i||$ for a stiff region and $||\boldsymbol{\xi}_i|| \leqslant ||\Delta \mathbf{y}_i||$ for non-stiff regions (see Shampine, 1979). Assuming that the parameters a_{ij} have an average value of 0·5 we obtain from (4.3) and (4.4) that

$$||h\Delta \mathbf{f}(\mathbf{y}_i)|| \leqslant 2^{i+1} \cdot \varepsilon, \quad i = 1(1)4.$$

For the computed solution $\mathbf{y}_{n+1}$ given by (1.3a) a bound on the error due to accumulated stage errors is

$$||\Delta \mathbf{y}_{n+1}|| \leqslant \varepsilon \cdot \sum_{i=1}^{s} 2^{i+1},$$

and we will require this to be less than the local error tolerance τ. Hence a conservative estimate of the magnitude of ε for an s-stage DIRK method is given by

$$\tau \Big/ \sum_{i=1}^{s} 2^{i+1}.$$

A similar analysis can be performed in order to assess the effect of accumulated errors on the local error estimation technique using embedding. This results in a comparable estimate for ε.

For implicit type Runge–Kutta methods the work involved per step is high and premature rejection of the stage calculation can lead to an overall increase in computational work. For this reason the number of iterations allowed to achieve convergence of a stage calculation is set at five. This has been selected experimentally. As well, an error reduction factor (ERF), based on function residuals can be computed in order to monitor the rate of convergence. If the ERF exceeds unity on two successive iterations then the iteration is abandoned and remedial action is taken. The most difficult stages to compute are those whose starting values require extensive extrapolation. In the case of the proposed methods these are stages 1 and 3. If convergence is achieved for these stages then the intermediate stages (2 and 4) are rapidly computed.

The failure of the iterative procedure via excess iterations or an unacceptable ERF can have two basic causes. Either the partial derivative information in the Jacobian is outdated and/or the starting values for the stage calculations are not sufficiently accurate. The latter is due mainly to an excessive steplength affecting the accuracy of the extrapolation. If storage requirements are not important (!) it is possible to retain a copy of the Jacobian to be used when the steplength is changed. Generally a change in steplength normally implies a new factorization of the matrix $(\mathbf{I} - h\gamma\mathbf{J})$ and hence a re-evaluation of $\mathbf{J}$ by finite differencing. If $\mathbf{J}$ has been evaluated within the previous three steps, the steplength is halved in order to improve convergence and $(\mathbf{I} - h\gamma\mathbf{J})$ is factorized, otherwise the Jacobian is re-evaluated and the steplength retained. In the case where no copy of $\mathbf{J}$ is stored, it is necessary to re-evaluate $\mathbf{J}$ when the steplength is altered.

In the case of a successful computation of the stage calculations and compliance with the local error tolerance, the maximum *ERF* during the step is monitored and if this value exceeds 0·2 then the Jacobian is updated on the next step regardless of whether the steplength has changed. Of major importance in DIRK codes is the necessity of providing good quality starting values for the iterative solution procedure. The code uses a combination of extrapolation and interpolation to produce the initial estimates. It is found that the quality of the estimates has a significant effect on code reliability and performance.

4.2 *Step Selection*

Local step selection is based on truncation error considerations, the general mechanism used to estimate the nth step for the pth order method is

$$h_{n,p} = h_{n-1,p}\left(\frac{c \cdot \tau}{L_e}\right)^{\frac{1}{p+1}},$$

where c is a constant in the range (0, 1) and L_e is the weighted local error estimate for the pth order method given by

$$L_e = \left\{\sum_{i=1}^{m} [(y_{n+1}^{(p+1)}(i) - y_{n+1}^{(p)}(i))/W_{y(i)}]^2\right\}^{\frac{1}{2}}.$$

The value of c is adjusted to give a conservative steplength change ($c = 0{\cdot}5$).

Steplength increases are subject to a bound B, such that a step increase is only considered if $h_n > B_{\min}(h_{n-1})$. This bound can be established by comparing the computational work involved in re-evaluation of the Jacobian plus LU-factorization of the iteration matrix $(\mathbf{I} - h\gamma\mathbf{J})$ as opposed to proceeding with the current steplength. Assuming dense matrix techniques and $O(m^2)$ operations in evaluating the equations with an average of two iterations for the convergence of the stage calculations the values of $B_{\min}$ are given in the following table.

TABLE 2

s \ m	5	10	20	50
1	2·0	3·5	6·6	16·5
2	1·5	2·2	3·8	8·7
3	1·3	1·8	2·9	6·2
4	1·25	1·6	2·4	4·9

Not only is a minimum steplength increase necessary but a bound needs to be placed on the maximum allowable steplength increase factor $B_{\max}$, under normal integration. This is currently set at 10. As well, it is necessary to reduce $B_{\max}$ to a much lower value of two for several steps after a convergence failure of the stage calculations or a failure in the local error estimation. This avoids "chattering" in the step-change mechanism and improves the code robustness on difficult problems.

The user of a code for dynamic simulation purposes does not want to be burdened with the selection of the initial steplength h_0 or for restarting the code after a discontinuity has been encountered. The only information available to the code for this purpose is the estimates of the initial slope of the differential variables. An approximation of h^P times the zeroth order error hf_i can be used to estimate the principal truncation error of the pth order method. This must be less than the required local error $y_i\tau$. To be conservative, we reduce the step estimate by τ, since nothing is known about the behaviour of the algebraic system at the point of discontinuity. This is because in some cases the algebraic system can control the rate of convergence of the stage calculation. The estimated steplength for restart is then given by

$$h_0 = \tau . \min_{1 \leq i \leq m} \left| \frac{y_i \tau}{f_i} \right|^{\frac{1}{p+1}},$$

which has proved quite successful in practice, and is similar to a device used in the explicit Runge–Kutta code RKF45 (Shampine & Watts, 1979).

4.3 *Order Selection for Variable-order Implementation*

The basis of order selection is the minimization of local work which requires an assessment of work per step (w_p) for a pth-order method with s stages. Hence

$$w_p = s_p/\eta_p,$$

where s_p is the number of stages for the pth order method and n_p is an efficiency factor to account for the difficulty in exploiting the higher order methods. The work per unit step at t_{n-1} is given by

$$w_{n,p} = s_p/\eta_p h_{n,p},$$

with $\eta_p = 1{\cdot}0, 0{\cdot}9, 0{\cdot}85$ ($p = 1, 2, 3$) being obtained heuristically. In this respect order changing is similar to several implementations of the backward differentiation formulae (Gear, 1971*a*). However, DIRK methods can start at high order and if enough accuracy is demanded it is advantageous to start at a higher order. Thus the user-set local error tolerance τ can be used to determine the initial starting order. For an error tolerance $\tau < 10^{-4}$ the starting order has been set at four otherwise the code commences with the second-order formula. Allowing order change at every step adds too much to the code overhead. So for a method of order p, a change in order is only considered after $K(p)$ steps. Numerical experiments suggest that K should be in the range 3–5.

5. Numerical Experience on some Differential–Algebraic Systems

Results are quoted for two variable step-variable order codes. One is based on the fully embedded formulae (2.1) and is named DIRKA. The other code is based on the approach using "triples" and is designated DIRKS. As well as these codes, results are given for each code when used in a fixed-order mode under the same implementation. These results are denoted by DIRKA i or DIRKS i, where i is the

fixed-order method. The problems used here are given in the Appendix. Other examples of more complex systems are given in Cameron (1981).

The results obtained are given in Tables 3 and 4, quoting the number of steps (NS), number of calls to the equations (NEQC), the number of factorizations (NFAC) and the number of Jacobians evaluated (NJ). Computation time (CPU) is given in milliseconds for a CDC Cyber 74 using an MNF5 compiler. The "Error" value quoted in Tables 3 and 4 is stated for each problem. As far as it was possible, the implementation aspects of both codes were identical, except in the prediction of starting values for the stage calculations.

Problem DAS 1 is initially non-stiff but increases in stiffness as the solution proceeds. It has a rapid transient in the interval (0·1, 1·0) where the algebraic variables are changing at 10 times the rate of the differential variables. The "Error" quoted is the relative error in y_1 at the end of the interval. For this problem the fixed-order methods DIRKA i display the expected behaviour in that the low-order method is more efficient on low accuracy requirements and vice versa. The variable-

TABLE 3
Problem DAS 1

Code	τ	NS	NEQC†	NFAC/NJ	CPU	Error
	10^{-2}	32	351	20/20	710	7·439E–5
DIRKA	10^{-3}	38	621	26/26	1230	4·665E–5
	10^{-4}	57	1016	32/32	2050	2·099E–6
	10^{-2}	30	364	21/21	700	1·469E–4
DIRKS	10^{-3}	48	746	30/30	1410	7·365E–6
	10^{-4}	70	1524	46/46	2880	2·234E–6
	10^{-2}	27	294	17/17	600	9·795E–5
DIRKA 2	10^{-3}	58	584	26/26	1160	1·117E–5
	10^{-4}	165	1513	53/53	3080	4·779E–6
	10^{-2}	44	455	27/27	840	3·911E–4
DIRKS 2	10^{-3}	104	931	49/49	1700	4·625E–6
	10^{-4}	321	2683	125/125	4940	9·857E–7
	10^{-2}	26	366	19/19	780	8·595E–4
DIRKA 3	10^{-3}	48	676	29/29	1350	1·577E–5
	10^{-4}	96	1337	44/44	2690	1·286E–6
	10^{-2}	22	334	19/19	640	2·042E–4
DIRKS 3	10^{-3}	46	602	30/30	1120	8·342E–7
	10^{-4}	91	1146	47/47	2150	1·366E–6
	10^{-2}	24	389	18/18	840	2·064E–5
DIRKA 4	10^{-3}	33	602	25/25	1200	4·809E–5
	10^{-4}	55	1017	31/31	2090	2·074E–6
	10^{-2}	24	514	22/22	1000	2·939E–4
DIRKS 4	10^{-3}	37	885	31/31	1660	2·171E–6
	10^{-4}	70	1617	43/43	3075	3·554E–6

† For DIRKA, NEQC includes the separate calls to the algebraic system made at the end of each step.

order implementation shows an average performance compared with the fixed-order results over the accuracy range. This is mainly due to code overhead in assessing order change as well as the delay in selecting the appropriate order method. The order-changing behaviour of the DIRKA code was as expected. High-order methods were chosen during the transient period with subsequent order reduction in the steady-state region. As the error tolerance was tightened a tendency to remain at the higher orders was observed.

The DIRKA code shows improved performance over DIRKS especially at higher accuracy. This is a result of the poor performance of the fourth-order DIRKS method which requires five-stage calculations per step and possesses a quadrature point well outside the current steplength (viz. $c_2 = -0{\cdot}7$). This has serious consequences with discontinuous problems as will be evident in problem DAS 2. Of the fixed-order formulae the DIRKS 3 formula performs the most satisfactorily. This has three-stage calculations within the steplength and no necessity for a trimming iteration on the algebraic variables at the end of the step. The reported solution accuracy is much smaller in some cases than was required. This is the result of taking the computed error at the end of the integration interval which is in the steady-state region. As well, the step-changing mechanism used in the code is conservative.

TABLE 4

Problem DAS 2

Code	τ	NS	NEQC	NFAC/NJ	CPU	Error
	10^{-2}	78	985	58/58	2020	2·154E–4
DIRKA	10^{-3}	112	1567	79/79	3250	1·704E–4
	10^{-4}	149	2395	101/101	5090	1·069E–4
	10^{-2}	78	639	58/26	1910	2·338E–4
DIRKA†	10^{-3}	112	1127	78/38	3080	1·696E–4
	10^{-4}	149	1845	97/47	4880	1·065E–4
	10^{-2}	75	888	52/52	1780	9·006E–5
DIRKS	10^{-3}	138	1807	97/97	3510	2·553E–4
	10^{-4}	204	3017	119/119	6080	2·468E–4
	10^{-2}	70	990	53/53	2110	1·849E–5
DIRKA 3	10^{-3}	107	1502	72/72	3160	1·146E–5
	10^{-4}	202	2230	110/110	6160	6·604E–7
	10^{-2}	63	850	50/50	1700	2·173E–4
DIRKS 3	10^{-3}	101	1344	71/71	2600	1·206E–4
	10^{-4}	190	2482	116/116	4920	3·495E–4
	10^{-2}	60	1007	52/52	2190	1·942E–4
DIRKA 4	10^{-3}	79	1375	62/62	2970	6·935E–6
	10^{-4}	122	2132	85/85	4610	2·830E–7
	10^{-2}	58	970	49/49	2010	2·132E–3
DIRKS 4	10^{-3}	90	1469	63/63	2990	1·357E–3
	10^{-4}	171	2786	103/103	5730	5·472E–4

† DIRKA using a copy of the Jacobian.

Problem DAS 2 displays sharp transients at the points of discontinuity in time. This is a multiple-event problem, characteristic of many process simulation problems. The "Error" value quoted is the relative error in liquid height at $t = 3$ h. Again, the DIRKA code was more efficient in handling this problem than DIRKS except at the lowest accuracy. Both codes used the auto-restart procedure for selecting the initial and restart steplengths via (4.5). This proved successful.

Table 4 also gives results for DIRKA using a copy of the Jacobian instead of necessarily recalculating the Jacobian at every change of step. In terms of equations calls, the use of a copy of the Jacobian gives a significant reduction. However, the corresponding reduction in computation time is not so significant. The use of only equation-calls as a measure of code performance can often be misleading. In this particular case the total number of iterations required increased when using a copy of the Jacobian due to slower convergence. As well, the savings generally decrease as accuracy is tightened. This is also confirmed on more complex simulation problems. Even so, if storage is not a problem, using a copy of the Jacobian can be worthwhile for large problems.

Of the fixed-order methods, the DIRKS 3 formula is better at lower accuracy whereas the fourth-order DIRKA formula shows superior performance at higher accuracy. The fourth-order DIRKS formula proves unsuitable for discontinuous problems not only because of its five stages but because of complications arising from the location of its quadrature points.

Numerical experience indicates that DIRK methods are adequately suited to the solution of differential–algebraic systems, especially multiple-event type problems like DAS 2. The approach to a variable step-variable order algorithm based on a fully embedded set appears more efficient than the use of "triples". Under certain conditions the variable order codes give a marginal improvement over fixed-order methods at the expense of increased code complexity. Within the fixed-order methods the third-order strongly S-stable algorithm (DIRKS 3) is very efficient for low accuracy. More work is required to improve the implementation aspects of DIRK codes especially in the area of the provision of more accurate starting values for the Newton-like iteration. Further consideration is needed of the stability properties of implicit methods suitable for unstable problems. This is one of the glaring areas of unreliablity in present implicit algorithms for stiff equations, and one that needs to be solved in order to produce reliable codes applicable to a wide range of process simulation problems.

The author would like to thank the editor and the referees for their invaluable help in the writing of this paper.

References

Alexander, R. 1977 Diagonally-Implicit Runge–Kutta methods for stiff ordinary differential equations. *SIAM J. num. Analysis* **14,** 1006–1022.

Beyer, W. H. 1978 *CRC Standard Mathematical Tables,* 25th Edition.

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

Cameron, I. T. 1981 Numerical solution of differential–algebraic systems in process dynamics. Ph.D. Thesis, University of London.

CASH, J. R. 1979 Diagonally-implicit Runge–Kutta formulae with error estimates. *J. Inst. Maths Applics* **24,** 293–302.

CASH, J. R. & LIEM, C. B. 1980 On the design of a variable order-variable step DIRK algorithm. *J. Inst. Maths Applics* **26,** 87–91.

COOPER, G. J. & SAYFY, A. 1979 Semi-explicit A-stable Runge–Kutta methods. *Maths Comput.* **33,** 541–556.

CROUZEIX, M. 1975 Sur l'approximation des équations différentielles operationnelles linéares par des méthodes de Runge–Kutta. Ph.D. Thesis, University of Paris 6.

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

GEAR, C. W. 1971*b* Simultaneous numerical solution of differential-algebraic equations. *IEEE Trans. Circuit Theory* **CT-18,** 89–95.

HAIRER, E. & WANNER, G. 1974 On the Butcher group and general multivalue methods. *Computing* **13,** 1–5.

NØRSETT, S. P. 1974*a* Semi-explicit Runge–Kutta Methods. Dept. Math, University of Trondheim Report 6/74.

NØRSETT, S. P. 1974*b* One step methods of Hermite type. *BIT* **14,** 63–77.

NØRSETT, S. P. 1975 C-polynomials for rational approximation to the exponential function. *Num. Math.* **25,** 39–56.

PROTHERO, A. & ROBINSON, A. 1974 On the stability and accuracy of one step methods for solving stiff systems of ordinary differential equations. *Maths Comput.* **28,** 145–162.

SHAMPINE, L. F. 1979 Implementation of Implicit Formulas for the solution of ODEs. SANDIA Report, SAND79–0694. Sandia Laboratories, Albuquerque, New Mexico 87185, U.S.A.

SHAMPINE, L. F. & WATTS, H.A. 1979 Practical Solution of Ordinary Differential Equations by Runge–Kutta Methods. SANDIA Report, SAND76–0585.

WOLFBRANDT, A. 1977 A Study of Rosenbrock processes with respect to order conditions and stiff stability. Report 77.01R, Dept. of Computer Science, Chalmers University of Technology, University of Göteborg.

Appendix

Problem DAS 1

This is a stiff test example taken from Gear (1971*b*) with four differential and four algebraic equations.

$$y_i' = s-(r-y_i)^2 - \sum_{y=1}^{4} b_{ij}y_j, \quad i = 1(1)4,$$

$$0 = y_5 - y_1 y_6,$$

$$0 = 2y_6 + y_6^3 - y_1 + y_7 - 1 - e^{-t},$$

$$0 = y_7 - y_8 + y_1 y_6,$$

$$0 = y_7 + y_8 + 5y_1 y_2,$$

with

$$r = \sum_{y=1}^{4} y_i/2, \qquad s = \sum_{y=1}^{4} (r-y_i)^2/2, \quad t \in (0, 10^3),$$

and initial conditions are

$$y_i(0) = -1, \quad i = 1(1)4,$$

$$y_5(0) = y_6(0) = 1, \qquad y_7(0) = -2, \qquad y_8(0) = -3,$$

also

$$\mathbf{B} = (b_{ij}) = \begin{bmatrix} 447{\cdot}5+\varepsilon & -452{\cdot}5+\varepsilon & -47{\cdot}5+\varepsilon & -52{\cdot}5-\varepsilon \\ -452{\cdot}5+\varepsilon & 447{\cdot}5+\varepsilon & 52{\cdot}5+\varepsilon & 47{\cdot}5-\varepsilon \\ -47{\cdot}5+\varepsilon & 52{\cdot}5+\varepsilon & 447{\cdot}5+\varepsilon & 452{\cdot}5-\varepsilon \\ -52{\cdot}5-\varepsilon & 47{\cdot}5-\varepsilon & 452{\cdot}5-\varepsilon & 447{\cdot}5+\varepsilon \end{bmatrix},$$

$\varepsilon = 0{\cdot}00025.$

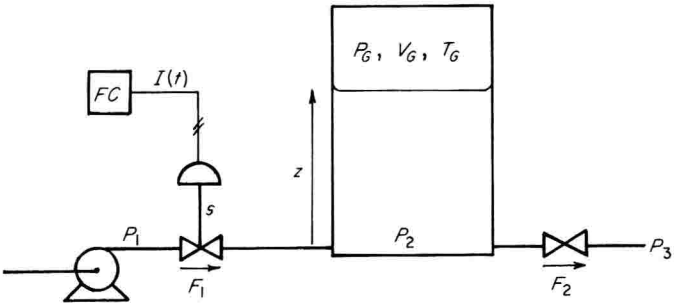

FIG. A1

Problem DAS 2

The system (Fig. A1) represents a tank being filled via a pump and control valve with the gas above the liquid subject to adiabatic compression. The system equations are:

Control valve

$$\frac{dy}{dt} = \frac{I(t)G}{\tau^2} - \frac{2\xi y}{\tau} - \frac{s}{\tau^2}$$

$$\frac{ds}{dt} = y$$

$$A_v = 0{\cdot}03 \exp(s/0{\cdot}28518).$$

Flow, height and pressure relations are:

$$\frac{dz}{dt} = (F_1 - F_2)/A$$

$$\frac{dT_G}{dt} = \frac{dz}{dt}\left(\frac{AP_G 200}{m_G C}\right)$$

$$F_1 = A_v C_v \sqrt{P_1 - P_2}$$

$$F_2 = C_v \sqrt{P_2 - P_3}$$

$$P_G = m_G R T_G / V_G 10^3$$

$$P_2 = P_G + \rho g z / 10^3$$

$$V_G = V_0 - Az,$$

where

s = stem position (0, 1)
$I(t)$ = signal to valve (0, 1)
G = gain (1)
τ = time constant of valve ($2{\cdot}77 \times 10^{-4}$ h)
ξ = damping ratio (0·8)
A_v = flow area (0, 1).

Hence there are four differential equations and six algebraic equations in the system.

$A = 12{\cdot}566\ \text{m}^2$
$\rho = 1000\ \text{kg/m}^3$
$V_0 = 201{\cdot}0619\ \text{m}^3$
$R = 8314$ Nm/kg mol K
$C = 30\,354$ J/kg mol K
$P_1 = 400$ kPa
$P_3 = 100$ kPa
$m_G = 8{\cdot}397$ kg mol
$C_v = 3{\cdot}4153\ \text{m}^3/\text{kPa}^{\frac{1}{2}}\text{h}$

A set of initial conditions is given by

$z = 0{\cdot}0$ m	$P_G = 100$ kPa
$T_G = 288$ K	$F_1 = 59{\cdot}154\ \text{m}^3/\text{h}$
$s = 1{\cdot}0$	$F_2 = 0{\cdot}0$
$y = 0{\cdot}0$	$V_G = 201{\cdot}0619\ \text{m}^3$
$A_v = 1{\cdot}0$	$P_2 = 100$ kPa.

The sequence of events in the simulation is

Time (h)	Process change
1·0	$I(t)$: 1·0 → 0·7
1·5	$I(t)$: 0·7 → 0·6
2·0	P_1: 400 kPa → 500 kPa
2·5	P_3: 100 kPa → 110 kPa
3·0	$I(t)$: 0·6 → 0·7
3·5	P_1: 500 kPa → 400 kPa P_3: 110 kPa → 100 kPa

$t \in (0, 10)$.