Numerical Algorithms 2 (1992) 337–350

Limits of parallelism in explicit ODE methods *

Robert D. Skeel

Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 West Springfield Avenue, Urbana, IL 61801-2987, USA

Hon-Wah Tam

Wolfram Research, Inc., 701 Budd Court, Campbell, CA 95008, USA

Communicated by C. Brezinski and J.C. Butcher
Received 20 July 1991; revised 17 February 1992

Numerical methods for ordinary initial value problems that do not depend on special properties of the system are usually found in the class of linear multistage multivalue methods, first formulated by J.C. Butcher. Among these the explicit methods are easiest to implement. For these reasons there has been considerable research activity devoted to generating methods of this class which utilize independent function evaluations that can be performed in parallel. Each such group of concurrent function evaluations can be regarded as a stage of the method. However, it turns out that parallelism affords only limited opportunity for reducing the computing time with such methods. This is most evident for the simple linear homogeneous constant-coefficient test problem, whose solution is essentially a matter of approximating the exponential by an algebraic function. For a given number of stages and a given number of saved values, parallelism offers a somewhat enlarged set of algebraic functions from which to choose. However, there is absolutely no benefit in having the degree of parallelism (number of processors) exceed the number of saved values of the method. Thus, in particular, parallel one-step methods offer no speedup over serial one-step methods for the standard linear test problem. Although the implication of this result for general nonlinear problems is unclear, there are indications that dramatic speedups are not possible in general. Also given are some results relevant to the construction of methods.

Subject classification: AMS(MOS) 65L05; CR G.1.7, G.1.0

* Work supported in part by National Science Foundation grants DMS 89 11410 and DMS 90 15533 and by US Department of Energy grant DOE DEFG02-87ER25026. Work of the second author was completed while at the University of Illinois.

1. Introduction

We consider explicit general-purpose numerical methods for solving the system of ordinary differential equations

$$y(a) = \eta,$$
$$y'(t) = f(y(t)), \quad t_0 \leqslant t \leqslant t_{\text{out}}.$$

(A nonautonomous ODE of the form $z'(t) = f(t, z(t))$ can always be written in the above autonomous form. Hence the above form involves no loss of generality.) In an earlier paper we announced some theoretical results concerning the use of parallelism. More specifically, we asserted that, for the model problem $y' = \lambda y$, one-step and multistep methods have no parallelism but that true multivalue methods have some parallelism, the degree of parallelism limited by the "number of saved values". This paper provides the technicalities and proofs promised in [17]. Also given are partial results relating to the construction of methods. In related work [24,19] we investigate the possibility of using parallelism to enlarge regions of absolute stability.

Gear [8,9] gives a survey on parallel ODE methods and classifies parallelism in solving ODEs into two main categories:

(1) parallelism across the method,

(2) parallelism across the system.

Parallelism across the method, or time, means that different parts of the method are executed on different processors. For explicit general linear methods this means the evaluation of f for different arguments on different processors. Parallelism across the method can be applied even to a single equation. Techniques in this category include block methods [2,5,15,19–23,25], frontal methods [13], so-called "stabbing" methods [9,14], Picard-like methods [1,11,16], and extrapolation methods [6]. Existing methods such as deferred correction [7] are also adaptable for time parallelism. *Parallelism across the system* refers to dividing the given equations into subsystems and assigning different subsystems to different processors. If the number of equations is large, this approach provides a larger opportunity of parallelism than parallelism across time, to the extent that the computation load can be balanced across processors.

While parallelism across the system has more potential for large scale parallelism, balancing the load between processors may be difficult. In some extreme cases, the bulk of the computation involves a single indivisible expensive stage, and the evaluation of each component subsystem is trivial after the computation of this stage. Even if parallelism across the system is efficient for a given problem, it may take a lot of user involvement to partition the system. Thus, if only for convenience, an average user may not consider parallelism across the system at all. Although parallelism across the method provides potentially only a small degree of parallelism, it has the advantage of no user programming burden. The user supplies, in the case of nonstiff ODEs, only a

subroutine to evaluate $f(y)$ for a given y. This is the same as what is required in conventional sequential ODE packages. The parallel software simply calls $f(y)$ a few times in parallel internally. For a maximum speedup, one would like to combine both parallelism across the method and system.

The test problem $y' = \lambda y$ has been very useful for studying numerical methods for ODEs and, in particular, has proved to have rather remarkable modeling abilities for stability. A good first step in the selection of method parameters is to optimize them for $y' = \lambda y$. Then the additional degrees of freedom that exist for the general ODE $y' = f(y)$ can be used to optimize the accuracy for the general ODE.

Most current explicit methods advance in time by means of a formula which for uniform stepsize is of the form

$$y_n + \alpha_1 y_{n-1} + \cdots + \alpha_k y_{n-k} = h\psi(y_{n-1}, \ldots, y_{n-k}; h).$$

Stetter [18] calls such methods *straight multistep methods*. Such methods are a special case of a general k-value method

$$z_n = Az_{n-1} + h\Phi(z_{n-1}; h),$$

where z_n is a set of k values needed for future steps, A is a $k \times k$ matrix, and Φ is a vector of k "increment" functions.

The remainder of the paper is organized as follows. Section 2 introduces the class of general linear methods due to Butcher, which includes most existing sequential and parallel methods. Section 3 shows *for the linear test problem* that straight multistep methods have no speedup due to parallelism and that for general multivalue methods having more processors than the number of saved solution values is useless. Section 4 introduces the stability polynomial and considers the number of degrees of freedom there are in a method when applied to the homogeneous constant-coefficient linear test equation and how this depends on the number of processors. The stability polynomial completely determines the algebraic function which approximates the exponential solution of the simple linear homogeneous constant-coefficient test problem. For a given number of stages and a given number of saved values, parallelism offers more free parameters and a somewhat enlarged set of algebraic functions from which to choose as long as the number of processors does not exceed the number of saved values of the method. Section 5 considers the construction of general linear methods from the stability polynomial.

If we consider the general nonlinear ODE, there is evidence that parallelism offers an advantage that is proportional to the desired accuracy. With an unlimited number of processors, the ultimate explicit method might be the classical Picard iteration with each iteration counting as one function evaluation. An experiment by Skeel [16] indicates that the speedup due to parallelism is just over one half the number of digits of accuracy. Perhaps this can be explained in terms of the elementary differentials which constitute the local truncation error.

If we have m stages, then we can with enough processors remove from the local truncation error any elementary differential whose corresponding rooted tree has at most m levels. Thus parallelism can eliminate those bushy trees in which there are "parallel" branches but can do nothing for the branchless trees which are represented by the linear test problem. If the desired accuracy is high, so is the optimal order and the proportion of elementary differentials with bushy trees. Another way of motivating this is as follows. The number of stages required by a sequential pth order explicit Runge–Kutta method seems to increase superlinearly with p but is no worse than about $p^2/4$. (Consider extrapolation applied to Gragg's modified explicit midpoint method with substepsize sequence $h/(2k)$, $k = 1, 2, \ldots, p/2$.) Suppose that the optimal order for d digits is $2d$. (This has been obtained from a back-of-the-envelope calculation and is not too different from the sometimes uttered rule of thumb that the order should be equal to the desired digits of accuracy.) The number of stages required by a parallel pth order explicit Runge–Kutta method is exactly p. Thus the speedup factor due to parallelism is roughly $p/4 = d/2$.

2. General linear methods

What is the most general "linear" method for p processors? We begin with k saved solution values that are passed on from the previous step. Within the step we perform m stages. Each stage involves up to q independent evaluations of f performed in parallel on separate processors – for arguments which are linear combinations of the saved values and previous function values evaluated within the current step. Let z_{n-1} denote the vector of the k saved values, and $Z_1, \ldots, Z_m$ the q-dimensional vectors of the internal stage values. Then an explicit general linear method (explicit *GLM*) is given by

$$Z_i = C_i z_{n-1} + h \sum_{j=1}^{i-1} L_{ij} F_j, \quad i = 1, 2, \ldots, m,$$

$$z_n = A z_{n-1} + h \sum_{j=1}^{m} B_j F_j, \tag{1}$$

where $F_i = f(Z_i)$ (componentwise), and the matrices C_i, L_{ij}, A, B_j are of dimensions $q \times k$, $q \times q$, $k \times k$, $k \times q$, respectively. Equation (1) can be expressed

$$\begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_m \\ z_n \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \\ A \end{bmatrix} z_{n-1} + h \begin{bmatrix} 0 & & & & \\ L_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ L_{m1} & \cdots & L_{m,m-1} & 0 & \\ B_1 & \cdots & B_{m-1} & B_m & \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{bmatrix}$$

or more compactly,

$$\begin{bmatrix} z_n \\ Z \end{bmatrix} = \begin{bmatrix} A & B \\ C & L \end{bmatrix} \begin{bmatrix} z_{n-1} \\ hF \end{bmatrix}. \tag{2}$$

Butcher [3] first proposed this general class of methods. The only difference in the case of parallelism is that for an explicit method instead of L being strictly lower triangular, it is now block strictly lower triangular. Also note we have defined a *stage* of a method to be a set of function evaluations performed in parallel and the *stage number* m of a method to be the number of such stages per step in the method. Formulation (1) includes Runge–Kutta methods, linear multistep methods, one-leg methods, cyclic methods, sequential and block predictor–corrector methods, frontal methods, and multi-block methods.

When (1) is applied to a linear test problem $y' = \lambda y$, we get $hF_i = \mu Z_i$ where $\mu = h\lambda$, and

$$z_n = A(\mu) z_{n-1},$$

where $A(\mu)$ is a $k \times k$ matrix with entries that are polynomials in μ of degree $\leqslant m$:

$$\begin{aligned} A(\mu) &= A + \mu B(I - \mu L)^{-1} C, \\ &= A + \mu BC + \mu^2 BLC + \cdots + \mu^m BL^{m-1} C. \end{aligned} \tag{3}$$

We call $A(\mu)$ the stability matrix of (1), because it determines the stability of the numerical solution.

To be useful, a method must have a starting procedure. This is of the same form as a GLM except that C and A are column vectors, so for the model problem we would have

$$z_0 = s(\mu)\eta$$

for some vector $s(\mu) = s_0 + s_1\mu + \cdots$ of dimension k having entries which are polynomials in μ. In addition there should be a "finishing" procedure [3,4] which would be of the form of a GLM with A and B as row vectors so that for the model problem the method would generate approximations

$$y_n := t(\mu)^{\mathrm{T}} z_n$$

to the solution values $y(t_n)$. For an efficient finishing procedure one would not want to do any function evaluations that would not be used anyway to advance the solution. Because the forward step procedure is applied repeatedly, its properties are of the greatest importance; so it is designed first and the starting and finishing procedures afterwards.

It seems natural to require that z_n be in the limit $h \to 0$ the same as the starting procedure applied to $y(t_n)$, and, hence, by considering the model problem we have

$$z_n = s(\mu) \exp(\lambda(t_n - t_0))\eta + O(h). \tag{4}$$

Substituting into (1), we get

$$As_0 = s_0. \tag{5}$$

Also it does not make sense to evaluate f at points Cz_n unless

$$Cz_n = ey(t_n) + O(h),$$

where e is the vector of all ones. Applying this to (4), we get

$$Cs_0 = e. \tag{6}$$

Equations (5) and (6) are called *preconsistency* conditions [4,10].

3. Simplification for the model problem

This section demonstrates that explicit (straight) multistep methods offer no speedup via parallelism for the model problem $y' = \lambda y$. This hints at the possibility that for general problems explicit Runge–Kutta and predictor–corrector methods do not offer much significant speedup from the use of parallelism. These results indicate that practical parallel ODE methods must be true multivalue methods. We further show that having more processors than the number of saved values offers no speedup for a general linear method.

It has been proposed [12] that some internal stages of specially designed explicit Runge–Kutta methods can be computed in parallel. Thus although the following Runge–Kutta method

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & 0 \\ a_{31} & 0 & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 & 0 \\ a_{51} & a_{52} & a_{53} & 0 & 0 \end{bmatrix}$$

has five function evaluations, with parallelism ($q = 2$) there are only three stages[1] in the sense of (1). The corollary to the following theorem shows that methods of this type give no improvement over sequential methods for the model problem.

THEOREM 1

For any explicit q-processor m-stage GLM where rank $B =: r < q$ there exists an explicit r-processor m-stage GLM which is equivalent for all problems of the form $y' = \lambda y$. Moreover, under a minor technical restriction given in the proof of the theorem we can make the r-processor method preconsistent, assuming that the q-processor method is.

[1] To put this in the form (1), one could duplicate the first function evaluation.

Proof

We can write

$$B = UV,$$

where U is k by r and V is r by mq. The solution given by such a method will be fully determined by its stability matrix which will, see (3), have the form

$$A(\mu) = A + \mu UW_1 + \cdots + \mu^m UW_m.$$

For the r-processor method choose the k by mr matrix $\bar{B} = U[0 \ldots 0\, \bar{V}_m]$ where $\bar{V}_m$ is to be determined and the r by r matrices $\bar{L}_{ij} = 0$, $j < i - 1$. Then the equations which must be satisfied by $\bar{L}_{21}, \ldots, \bar{L}_{m,m-1}$, $\bar{V}_m$, and $\bar{C}$ are

$$\bar{V}_m \bar{L}_{m,m-1} \cdots \bar{L}_{m-j+2,m-j+1} \bar{C}_{m-j+1} = W_j, \quad j = 1, 2, \ldots, m. \tag{7}$$

Then, if we want preconsistency and if $W_i e \neq 0$ for all i, choose nonsingular r by r matrices $\bar{V}_m, \bar{L}_{m,m-1}, \ldots, \bar{L}_{21}$ that satisfy $\bar{V}_m e = W_m s_0$, $\bar{L}_{m,m-1} e = \bar{V}_m^{-1} W_{m-1} s_0$, $\bar{L}_{m-1,m-2} e = \bar{L}_{m,m-1}^{-1} \bar{V}_m^{-1} W_{m-2} s_0$, etc. Otherwise, just choose them to be nonsingular. It is enough now to choose $\bar{C}$ to satisfy eqs. (7). □

For straight multistep methods B has rank 1.

COROLLARY 1

More than 1 processor offers no advantage for one-step and (straight) multistep methods for the problem $y' = \lambda y$.

THEOREM 2

Let

$$A(\mu) = A + \mu A_1 + \cdots + \mu^m A_m$$

be a $k \times k$ matrix where $As_0 = s_0$. Then there exists a preconsistent k-processor m-stage k-value GLM with stability matrix $A(\mu)$ and preconsistency vector s_0.

Proof

Choose the subdiagonal blocks of L to be any nonsingular matrix and let the other nonzero blocks of L be anything. Let C_1 be any nonsingular matrix satisfying $C_1 s_0 = e$ and for $j > 2$ let C_j be any matrix satisfying $C_j s_0 = e$. Then B must be chosen to satisfy the system

$$B[C, LC, \ldots, L^{m-1}C] = [A_1, A_2, \ldots, A_m].$$

The coefficient matrix of B is block lower triangular with nonsingular diagonal blocks, so we can uniquely determine B. □

COROLLARY 2

Given a preconsistent q-processor k-value GLM with $q > k$, there exists a preconsistent k-processor k-value GLM which is equivalent for the model problem $y' = \lambda y$.

Proof

The stability matrix of the q-processor method satisfies the hypothesis of the theorem 2. □

Hence, for the standard linear test problem the parallel method with $q > k$ offers no advantage over the one using only k processors. Therefore in deriving explicit parallel ODE methods there is a diminished advantage in having more processors than the number of saved solution values.

4. The stability polynomial and independent degrees of freedom

The stability polynomial $\Psi(\xi, \mu)$ of (1) is defined to be the characteristic polynomial

$$\Psi(\xi, \mu) = \det[\xi I - A(\mu)] = \xi^k + \sum_{j=1}^{k} \alpha_j(\mu)\xi^{k-j}, \tag{8}$$

where the $\alpha_j(\mu)$'s are polynomials in μ.

From $\Psi(A(\mu), \mu) = 0$ it follows that

$$z_n + \alpha_1(\mu)z_{n-1} + \cdots + \alpha_k(\mu)z_{n-k} = 0,$$

and applying $t(\mu)^{\mathrm{T}}$, we have

$$y_n + \alpha_1(\mu)y_{n-1} + \cdots + \alpha_k(\mu)y_{n-k} = 0.$$

Hence, the method is primarily characterized by its stability polynomial, if roundoff error is neglected.

The stability polynomials of general linear methods have a certain form.

LEMMA 1

The stability polynomial (8) of a general linear method satisfies

$$\deg \alpha_j(\mu) \leqslant \min\{jm, qm\}.$$

Proof

From (3),

$$\begin{aligned} A(\mu) &= A + \mu B(I - \mu L)^{-1}C \\ &= A + \mu BC + \mu^2 BLC + \cdots + \mu^m BL^{m-1}C, \end{aligned}$$

it is obvious that the degree of $\alpha_j(\mu)$ in (8) is $\leqslant jm$. Using the identity

$$\begin{bmatrix} \xi I - A - \mu B(I - \mu L)^{-1}C & 0 \\ -C & I - \mu L \end{bmatrix}$$

$$= \begin{bmatrix} I & \mu B(I - \mu L)^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} \xi I - A & -\mu B \\ -C & I - \mu L \end{bmatrix},$$

we have

$$\det\left[\xi I - A - \mu B(I - \mu L)^{-1}C\right] = \det\begin{bmatrix} \xi I - A & -\mu B \\ -C & I - \mu L \end{bmatrix}. \tag{9}$$

Since there are at most qm columns in μB and $I - \mu L$, the degree of $\alpha_j(\mu)$ in $\Psi(\xi, \mu)$ is $\leqslant qm$ and the lemma follows. □

5. Construction of GLMs from the stability polynomial

Let

$$\Psi(\xi, \mu) = \xi^k + \sum_{j=1}^{k} \alpha_j(\mu)\xi^{k-j} \tag{10}$$

be a given polynomial in ξ and μ such that the degree of each polynomial $\alpha_j(\mu)$ is less than or equal to $\min\{jm, qm\}$ for some q and m. It is of interest to know whether there always exists an m-stage q-processor method whose stability polynomial is given by $\Psi(\xi, \mu)$. For $k = 1$

$$\Psi(\xi, \mu) = \xi + \alpha_{10} + \alpha_{11}\mu + \cdots + \alpha_{1m}\mu^m$$

$$= \det\left[\begin{array}{c|ccccc} \xi + \alpha_{10} & \mu\alpha_{11} & \mu\alpha_{12} & & & \mu\alpha_{1m} \\ \hline -1 & 1 & & & & \\ 0 & -\mu & 1 & & & \\ \vdots & & \ddots & \ddots & & \\ 0 & & & -\mu & 1 & \end{array}\right],$$

and comparison with (9) shows the GLM. Also for $m = 1$, $q = 1$, we have

$$\Psi(\xi, \mu) = \det\left[\begin{array}{cccc|c} \xi + \alpha_{10} & \alpha_{20} & \cdots & \alpha_{m0} & -\mu \\ -1 & \xi & & & 0 \\ & \ddots & \ddots & & \vdots \\ & & -1 & \xi & 0 \\ \hline \alpha_{11} & \alpha_{21} & \cdots & \alpha_{m1} & 1 \end{array}\right],$$

which again shows the GLM. More generally, however, the existence of a GLM requires at least a consistency criterion on $\Psi(\xi, \mu)$. Moreover, even if a method does exist, it may not be easy to find.

Consider the case $q = k = 2$. Let Π_m denote the space of polynomials of degree m or less with real coefficients. Suppose that we wish to find a GLM with stability polynomial

$$\Psi(\xi, \mu) = \xi^2 + \alpha_1(\mu)\xi + \alpha_2(\mu), \tag{11}$$

where $\alpha_1(\mu) \in \Pi_m$ and $\alpha_2(\mu) \in \Pi_{2m}$ are given. Necessary for the existence of a GLM is the existence of an appropriate stability matrix, and only this question is considered here. Finding the stability matrix of an m-stage 2-processor method is equivalent to finding $a(\mu)$, $b(\mu)$, $c(\mu)$ and $d(\mu) \in \Pi_m$ such that

$$\det\begin{bmatrix} \xi - a(\mu) & -b(\mu) \\ -c(\mu) & \xi - d(\mu) \end{bmatrix} = \xi^2 + \alpha_1(\mu)\xi + \alpha_2(\mu)$$

$$\Leftrightarrow \left[\xi - \frac{a(\mu)+d(\mu)}{2}\right]^2 - \left[\frac{a(\mu)-d(\mu)}{2}\right]^2 - b(\mu)c(\mu) \tag{12}$$

$$= \left[\xi + \frac{\alpha_1(\mu)}{2}\right]^2 + \alpha_2(\mu) - \left[\frac{\alpha_1(\mu)}{2}\right]^2.$$

Equation (12) is equivalent to

$$a(\mu) + d(\mu) = -\alpha_1(\mu), \tag{13}$$

and

$$[a(\mu) - d(\mu)]^2 + 4b(\mu)c(\mu) = Q(\mu), \tag{14}$$

where $Q(\mu)$ is defined to be $\alpha_1(\mu)^2 - 4\alpha_2(\mu)$. If $\Psi(\xi, \mu)$ in (11) satisfies the preconsistency condition

$$\Psi(1, 0) = 0$$

and the root condition, we are able to prove the following lemma.

LEMMA 2

Let (11) be a given polynomial such that $\alpha_1(\mu) \in \Pi_m$, $\alpha_2(\mu) \in \Pi_{2m}$, and $\Psi(\xi, 0)$ has a simple root at $\xi = 1$. Then there exist $a(\mu)$, $b(\mu)$, $c(\mu)$, and $d(\mu) \in \Pi_m$ such that (12) holds.

Proof

The consistency condition implies that

$$0 = 1 + \alpha_1(0) + \alpha_2(0).$$

Therefore $Q(\mu) = \alpha_1(\mu)^2 - 4\alpha_2(\mu)$ satisfies $Q(0) = [1 - \alpha_2(0)]^2 > 0$. It is possible to show that $Q(\mu) = [1 - \alpha_2(0) + \mu w(\mu)]^2 + \mu^{m+1} v(\mu)$ for some $w(\mu)$, $v(\mu) \in \Pi_{m-1}$ – solve first for $w(\mu)$, term by term. Hence, eqs. (13) and (14) are satisfied by setting

$$a(\mu) = \tfrac{1}{2}[1 - \alpha_2(0) + \mu w(\mu) - \alpha_1(\mu)],$$

$$b(\mu) = \tfrac{1}{2}\mu^m,$$

$$c(\mu) = \tfrac{1}{2}\mu v(\mu),$$

$$d(\mu) = \tfrac{1}{2}[-1 + \alpha_2(0) - \mu w(\mu) - \alpha_1(\mu)],$$

which is equivalent to constructing a stability matrix with stability polynomial $\Psi(\xi, \mu)$. □

As an example, a stability polynomial with $k = q = 2$, $m = 1$, and giving third order accuracy can be written

$$\xi^2 - \left(1 + \alpha + \frac{7-\alpha}{3}\mu\right)\xi + \alpha + \frac{4+2\alpha}{3}\mu + \frac{5+\alpha}{6}\mu^2.$$

Fourth order accuracy is possible only with $\alpha = -17$, which violates zero-stability. Evidence [24] suggests that for $1/3 \leqslant \alpha \leqslant 1$ the stability region is "optimal". Within this range accuracy is optimal for the linear test problem for $\alpha = 1/3$. The stability polynomial for $\alpha = 1/3$ can be realized by the 2-processor method [19, p. 54]

$$\begin{bmatrix} A & B \\ C & L \end{bmatrix} = \left[\begin{array}{cc|cc} 16/9 & -7/9 & 65/36 & -43/36 \\ -1/9 & 10/9 & 37/36 & 1/36 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right].$$

This is shown to be third order accurate for the general nonlinear ODE, if z_n is regarded as an approximation to $[y(t_{n+1/2}),\ y(t_n)]^{\mathrm{T}}$.

Lemma 2 states that for any polynomial (10) with $q = k = 2$ satisfying the consistency condition, there exists the stability matrix of a GLM possessing $\Psi(\xi, \mu)$ as its stability polynomial. If we wish to construct the stability matrix for the case $q = k = 3$, we must find $a_{ij}(\mu) \in \Pi_m$ such that

$$\det\begin{bmatrix} \xi - a_{11}(\mu) & -a_{12}(\mu) & -a_{13}(\mu) \\ -a_{21}(\mu) & \xi - a_{22}(\mu) & -a_{23}(\mu) \\ -a_{31}(\mu) & -a_{32}(\mu) & \xi - a_{33}(\mu) \end{bmatrix}$$
$$= \xi^3 + \alpha_1(\mu)\xi^2 + \alpha_2(\mu)\xi + \alpha_3(\mu),$$

where $\alpha_1(\mu) \in \Pi_m$, $\alpha_2(\mu) \in \Pi_{2m}$, and $\alpha_3(\mu) \in \Pi_{3m}$. In order to determine the $a_{ij}(\mu)$'s, we need to solve $6m + 3$ equations for $9m + 9$ unknown coefficients, with the constraint that all these unknown coefficients are real. In general, if $q \geqslant k$, we have many more unknowns than equations, so we conjecture that *for any given polynomial $\Psi(\xi, \mu)$ of the form (10) with some minor consistency restrictions there exists a k-value m-stage q-processor stability matrix having $\Psi(\xi, \mu)$ as its stability polynomial.*

References

[1] A. Bellen, Parallelism across the steps for difference and differential equations, in: *Numerical Methods for Ordinary Differential Equations*, eds. A. Bellen, C.W. Gear and E. Russo,

Proceedings, L'Aquila, 1987, Lecture Notes in Mathematics no. 1386 (Springer, Berlin, 1989) pp. 22–35.
[2] L.G. Birta and O. Abou-Rabia, Parallel block predictor–corrector methods for ODE's, IEEE Trans. Comput. C-36 (1987) 299–311.
[3] J.C. Butcher, On the convergence of numerical solutions to ordinary differential equations, Math. Comp. 20 (1966) 1–10.
[4] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations – Runge–Kutta and General Linear Methods* (Wiley, New York, 1987).
[5] M.T. Chu and H. Hamilton, Parallel solution of ODE's by multi-block methods, SIAM J. Sci. Statist. Comput. 8 (1987) 342–353.
[6] P. Deuflhard, Recent progress in extrapolation methods for ordinary differential equations, SIAM Rev. 27 (1985) 505–536.
[7] L. Fox, Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, Proc. Roy. Soc. London Ser. A 190 (1947) 31–59.
[8] C.W. Gear, The potential for parallelism in ordinary differential equations, in: *Computational Mathematics II*, ed. S.O. Fatunla (Boole Press, Dublin, 1987) pp. 33–48.
[9] C.W. Gear, Parallel methods for ordinary differential equations, Calcolo: Quart. Num. Anal. Theory Comput. 25 (1989) 1–20.
[10] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Non-Stiff Systems* (Springer, Berlin, 1987).
[11] S. Horiguchi, Y. Kawazoe and H. Nara, A parallel algorithm for the integration of ordinary differential equations, *Proc. Int. Conf. on Parallel Processing* (1984) pp. 465–469.
[12] K.R. Jackson and S.P. Norsett, The potential for parallelism in Runge–Kutta methods. Part I: RK formulas in standard form, in preparation.
[13] W.L. Miranker and W. Liniger, Parallel methods for the numerical integration of ordinary differential equations, Math. Comp. 21 (1967) 303–320.
[14] J. Nievergelt, Parallel methods for integrating ordinary differential equations, Comm. ACM 7 (1964) 731–733.
[15] L.F. Shampine and H.W. Watts, Block implicit one-step methods, Math. Comp. 23 (1969) 731–740.
[16] R.D. Skeel, Waveform iteration and the shifted Picard splitting, SIAM J. Sci. Statist. Comput. 10 (1989) 756–776.
[17] R.D. Skeel and H.-W. Tam, Potential for parallelism in explicit linear methods, *Proc. IMA Conf. on Computational ODEs*, eds. J.R. Cash and I. Gladwell (Oxford Univ. Press), to appear.
[18] H.J. Stetter, *Analysis of Discretization Methods for Ordinary Differential Equations* (Springer, Berlin, 1973).
[19] H.-W. Tam, Parallel methods for the numerical solution of ordinary differential equations, Univ. of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-89-1516 (1989); also, Ph.D. thesis.
[20] H.-W. Tam, Parallel predictor–corrector block methods for ordinary differential equations, *Proc. 4th SIAM Conf. on Parallel Processing for Scientific Computing*, eds. J. Dongarra, P. Messina, D. Sorensen and R. Voigt (SIAM, Philadelphia, 1990) pp. 204–209.
[21] H.-W. Tam, A new parallel algorithm for the numerical solution of ordinary differential equations, *Proc. IMA Conf. on Computational ODEs*, eds. J.R. Cash and I. Gladwell (Oxford Univ. Press), to appear.
[22] H.-W. Tam, One-stage parallel methods for the numerical solution of ordinary differential equations, SIAM J. Sci. Statist. Comput 13 (1992), to appear.

[23] H.-W. Tam, Two-stage parallel methods for the numerical solution of ordinary differential equations, SIAM J. Sci. Statist. Comput 13 (1992), to appear.
[24] H.-W. Tam and R.D. Skeel, Stability of parallel explicit ODE methods, manuscript (1991).
[25] P.B. Worland, Parallel methods for the numerical solution of ordinary differential equations, IEEE Trans. Comput. C-25 (1976) 1045–1048.