Applied Numerical Mathematics 11 (1993) 5–25
North-Holland

APNUM 354

Parallel methods for initial value problems

Kevin Burrage

Centre for IAMPC, Department of Mathematics, The University of Queensland, Brisbane, Australia

Abstract

Burrage, K., Parallel methods for initial value problems, Applied Numerical Mathematics 11 (1993) 5–25.

As scientific technology becomes increasingly more sophisticated, the production of more data and/or the modelling of more complex systems requires computers with ever-increasing computational power. In order to cope with this situation numerical algorithms have to be developed which allow the distribution of both data and code segments over large numbers of processors with the hope that problems that were insoluble in a sequential environment because of either (or both) accuracy and size constraints can now be solved in a parallel environment.

This paper will present a review of recently developed techniques in the area of parallel numerical methods for initial value problems. It will focus mainly on two different approaches—parallelism across time and parallelism across space—but will also consider special techniques developed for certain classes of problems.

Keywords. Differential equations; waveform relaxation; prediction–correction.

1. Introduction

In recent years considerable attention has been given to the development of efficient parallel methods for the numerical solution of initial value problems (IVPs) of the form

$$\begin{aligned} &y'(t) = f(t, y(t)), \quad f:[t_0, T] \times \mathbb{R}^m \to \mathbb{R}^m, \\ &y(t_0) = y_0. \end{aligned} \tag{1.1}$$

So far it would be fair to say that limited progress has been made because of the fact that there is no natural parallelism across time when solving IVPs. However, the framework for much of the analysis of a large number of proposed methods has now been laid and it is hoped that this work will soon bear fruit and that some parallel production codes will be developed in the near future.

In order to give an idea of the magnitude of some of the problems that have to be dealt with, consider the problem of the modelling of long-range transport of air pollutants in the atmosphere (see, for example, Zlatev [73] and Zlatev and Berkowicz [75]). If $c_s(\theta, t)$ is the

Correspondence to: K. Burrage, Centre for IAMPC, Department of Mathematics, The University of Queensland, Brisbane, Australia.

concentration of the sth pollutant at space point θ and if $u(\theta, t)$, $v(\theta, t)$ and $w(\theta, t)$ are wind velocities along the x, y, and z coordinate axes, then this phenomenon can be modelled as

$$\frac{\partial c_s}{\partial t} = -\left(\frac{\partial(uc_s)}{\partial x} + \frac{\partial(vc_s)}{\partial y} + \frac{\partial(wc_s)}{\partial z}\right) + \frac{\partial(K_x \partial c_s/\partial x)}{\partial x} + \frac{\partial(K_y \partial c_s/\partial y)}{\partial y} + \frac{\partial(K_z \partial c_s/\partial z)}{\partial z} + E_s(\theta, t) - (k_{1s} + k_{2s})c_s(\theta, t) + R_s(c_1, \ldots, c_q), \quad s = 1, \ldots, q. \tag{1.2}$$

Here $E_s(\theta, t)$ represents the emission at space point θ and at time t for the sth pollutant, k_{1s} and k_{2s} are the dry and wet deposition coefficients, respectively, for the sth pollutant, $K(\theta,t)$ represents the diffusion coefficients along the three coordinate axes, and R_s describes the chemical reactions concerning the sth compound. (See Zlatev [74] for a more detailed exposition.)

This model, with $q = 29$, has been studied at the Danish Air Pollution Laboratory. Zlatev [74] notes that four relatively independent processes are associated with this model, namely advection, diffusion, deposition, and emission with chemistry. If the method of lines is used to solve this problem, then four systems of IVPs are generated which are to be solved cyclically at each time integration step. If a $32 \times 32 \times 9$ equidistant grid is specified for this three-dimensional problem, then each system of IVPs contains 267,264 equations which are to be solved at each time step. In addition, this model has to be solved over a long time scale in order to study monthly and seasonal variations of the concentrations. If greater accuracy is required, the spatial grid needs to be refined and this leads to even larger systems of equations.

Such problems are not solvable in reasonable time on a serial machine because of their magnitude. However, parallelism can also be exploited in many other ways. For example, if solutions are needed in real time, if there is a large period of integration, if parameter fitting is to be performed which requires repeated integration, and if function evaluations are expensive.

One very natural way to exploit parallelism associated with such large systems is by the splitting of the evaluation of the function components amongst the available processors. A second source of parallelism arises if the problem to be solved is stiff. Stiffness is a very difficult property to characterize explicitly but problems that are stiff typically arise from models which have widely differing time components. In particular, the technique of the method of lines applied to parabolic partial differential equations usually results in a stiff system if the spatial grid is moderately fine-grained. If a problem is stiff and nonlinear, then explicit methods are generally not suited because of unnatural restrictions on the stepsize and so implicit methods must be used. This means the solution of large systems of nonlinear equations at each time step. Hence codes based on the concept of BLAS for solving linear systems of equations in a parallel environment can be exploited at this stage (see the package LAPACK [20], for example).

These two natural ways of exploiting parallelism suggest that it is not possible to develop parallel algorithms in isolation from the parallel environment. Currently, there are no general-purpose parallel scientific libraries and while there have been some attempts at creating codes which can be ported between different parallel environments this has only focussed on the basic elements of array manipulation (BLAS).

It is not the intention of this article to spend much time discussing parallel environments—because of the complexity of the topic and its rather hybrid nature. However, four very basic types of parallelism can be identified (see Flynn [23], for example). Flynn's taxonomy is:

- SISD: single instruction stream, single data stream—as typified by the conventional sequential von Neumann architecture;
- SIMD: single instruction stream, multiple data stream—as typified by the MasPar series. Thousands of low-performance, small-memory processors are controlled in lockstep by a central controller.
- MISD: multiple instruction stream, single data stream—as typified by vector supercomputers such as the CRAY series. The computer paradigm for this process is called pipelining in which data proceeds sequentially in a pipeline through a number of processing stages.
- MIMD: multiple instruction stream, multiple data stream—as typified by a transputer network. Here the processors are independent, each executing its own sets of instructions on its own set of data. MIMD machines may share a common memory or be distributed with each processor having its own local memory.

It should be seen from this brief description that the architecture can and does have great impact on any parallel algorithm even at such basic levels as the BLAS. However, there is not sufficient space in this article to explore the relationship between environment and algorithm further and for the most part the discussion on parallel algorithms for IVPs will focus on an analytic discussion of the method's suitability as a parallel algorithm (except in some special cases). Indeed, it should be remarked that the Flynn taxonomy is now archaic. New architectures are being proposed which are hybrid in nature having elements from many different categories (see the CM5 by Thinking Machines, for example). Furthermore, there is an attempt by a number of people such as Valiant [65] to formulate a single parallel computing paradigm based on the concept of excess parallelism and fast communication which may be significant in terms of developing a coherent approach to parallel computing.

Finally, it should be noted that the implementational questions that need to be addressed in the development of a parallel production code for the numerical solution of differential equations (and, indeed, many other application areas) are considerably more complex than in the sequential case. Such facets that must be considered include the automatic load balancing of the work amongst processors, the avoidance of non-determinacy and deadlock, whether synchronous or asynchronous communication should be utilised and the avoidance of communication bottlenecks, and finally the gathering of appropriate performance statistics and suitable test problems. In some ways this last point is the most important of all. It does not seem appropriate to evaluate parallel codes on the same set of test problems used in the evaluation of sequential codes (such as DETEST, for example) because usually these problems are small in magnitude, or do not have large periods of integration, etc. Also it is important that the ubiquitous performance measure known as speed-up is interpreted in the correct way. It is not appropriate, and in fact is quite misleading, to base speed-up statistics on the comparison of the proposed code running on p processors and on one processor, since the one-processor implementation may be exceedingly inefficient. Rather the comparison should be made between the execution time of the parallel algorithm and the fastest extant sequential algorithm running on the same machine.

The outline of the rest of this paper is as follows. In Sections 2 and 3 the concepts of parallelism across the method and parallelism across the system will be introduced with Section 2 focussing on the former and Section 3 on the latter. In Section 4 novel parallel techniques for certain classes of problems will be discussed and the paper will conclude in Section 5 with some comments on future directions.

2. Parallelism across the method

Gear [26], in a seminal paper, proposed two different categories for developing parallel techniques for IVPs based on:

- parallelism across the method,
- parallelism across the system.

Algorithms that fall into the first category include those that exploit concurrent function evaluations within a step and other techniques (such as the parallel solution of linear recurrences) which solve simultaneously over a large number of steps; while algorithms of the second type include the recently developed approach of waveform relaxation (see White et al. [72], for example). Techniques based on parallelism across the method will be considered in Section 2.

2.1. Nonstiff direct methods

One way of exploiting parallelism across the method is to perform several function evaluations concurrently on different processors. This approach is only possible for multistage methods such as Runge–Kutta methods.

The class of s-stage Runge–Kutta methods for solving (1.1) is given by

$$\begin{aligned} Y_i &= y_n + h\sum_{j=1}^{s} a_{ij} f(t_n + c_j h, Y_j), \quad i = 1, \ldots, s, \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j f(t_n + c_j h, Y_j) \end{aligned} \tag{2.1}$$

and can be characterized by the Runge–Kutta tableau

$$\begin{array}{c|c} c & A \\ \hline & b^{\mathrm{T}} \end{array}$$

where

$$b^{\mathrm{T}} = (b_1, \ldots, b_s), \qquad A = (a_{ij})_{i,j=1}^{s},$$
$$c = (c_1, \ldots, c_s)^{\mathrm{T}}, \qquad c = Ae, \qquad e = (1, \ldots, 1)^{\mathrm{T}}.$$

Runge–Kutta methods belong to a more general class of methods called multivalue methods characterized by s internal stages and r updated stages. This class can be represented as

$$\begin{aligned} Y_i &= \sum_{j=1}^{r} a_{ij}^{(1)} y_j^{(n)} + h \sum_{j=1}^{s} b_{ij}^{(1)} f\left(t_n + c_j h, Y_j\right), \qquad i = 1, \ldots, s, \\ y_i^{(n+1)} &= \sum_{j=1}^{r} a_{ij}^{(2)} y_j^{(n)} + h \sum_{j=1}^{s} b_{ij}^{(2)} f\left(t_n + c_j h, Y_j\right), \quad i = 1, \ldots, r \end{aligned} \tag{2.2}$$

and is characterized by the tableau

$$\begin{array}{c|c} A_1 & B_1 \\ \hline A_2 & B_2 \end{array}$$

Here the Y_i represent approximations within the step to the solution at $t_n + c_i h$, $i = 1, \ldots, s$, while the $y_i^{(n)}$, $i = 1, \ldots, r$, carry all the updating information.

A Runge–Kutta method (respectively, a multivalue method) is said to be explicit if A (respectively B_1) is strictly lower triangular, otherwise it is implicit. If A (respectively B_1) is lower triangular, then the method is said to be diagonally implicit. An explicit multistage method is essentially sequential because in order to compute Y_{i+1}, say, $f(Y_1), \ldots, f(Y_i)$ are needed. However, in some cases two or more stages can be performed in parallel if there is an appropriate subset of the coefficients of the strictly lower triangular matrix A which are zero. This approach has been formalized by Iserles and Nørsett [31] who attempt to exploit the structure of the Runge–Kutta matrix using a diagraph analysis of this matrix. Their conclusion is that there is little to be gained in this respect.

As a consequence of this work, the concept of a block explicit method has been introduced (see Jackson and Nørsett [32], for example,) in which the Runge–Kutta matrix can be written in block lower triangular form. If the number of such blocks is k and the number of internal stages in the largest block is p, then such a method is suitable for a p-processor machine in which the p internal solution values can be computed concurrently within a block. In order to spread the work evenly (load balancing) it makes sense to have all blocks approximately the same size. However, taken as a whole, this approach is still sequential and appears not be of great value because Jackson and Nørsett [32] have shown that the maximum order of such a method is k and so is independent of p (the number of processors).

Similar results exist for more general multistage methods, so that in general this block value approach is of limited value. However, another direct approach through which parallelism can be exploited is via the concept of extrapolation. For example, Burrage and Plowman [16] and Nørsett [53] have considered a parallel implementation of the smoothed midpoint rule. The basic idea here is that any differential method can be applied over a time interval $[t_n, t_n + H]$ with a series of differing constant stepsizes $h_i = H/n_i$, $i = 1, \ldots, p$. Each n_i in the stepsize sequence is an integer and there are a number of possible ways of generating the n_i such as through doubling or by a harmonic Romberg sequence. If the underlying method has an even power series expansion of the stepsize in the global error, then Richardson extrapolation can be performed to generate high-order methods of any desired accuracy. An efficient parallel implementation requires the grouping of the p methods to ensure that each processor has approximately the same amount of work. This approach is well-suited for a parallel implemen-

tation when the problem size is large or function evaluations are very costly but again the parallelism is strictly limited.

2.2. Nonstiff iterative methods

One way of circumventing the somewhat negative features associated with parallelising block explicit Runge–Kutta methods is to consider a predictor–corrector approach. Predictor–corrector methods, which can be based on any appropriate class of discrete methods such as linear multistep, Runge–Kutta, or multivalue methods, can be used in the solution of stiff or nonstiff equations depending on the nature of the iteration.

Predictor–corrector methods were originally proposed by Moulton [47] in 1926 in conjunction with Adams methods and extended by Rosser [60] and Shampine and Watts [61] as examples of block methods. The underlying feature of a block method is that each application of the method generates a collection of approximations to the solution within a block. In a parallel environment individual processors can compute, independently, the approximation values to the solution within the block. Thus if Y_{n+1} represents a vector of solution values $y_{n+c_1}, \ldots, y_{n+c_s}$ and $F(Y_{n+1})$ represents the vector of derivatives $f(y_{n+c_1}), \ldots, f(y_{n+c_s})$, where c_s may equal 1 if the last value is used as an output point, then an s-processor implementation would hope to gain a speed-up of at most s by partitioning the problem and the method so that at each correction each processor simultaneously computes a subset of approximations.

An example of this approach is the two-point, fourth-order block method of Shampine and Watts [61] given by

$$y^{\mathrm{p}}_{n+1} = \tfrac{1}{3}(y^{\mathrm{c}}_{n-2} + y^{\mathrm{c}}_{n-1} + y^{\mathrm{c}}_{n}) + \tfrac{1}{6}h(3f^{\mathrm{c}}_{n-2} - 4f^{\mathrm{c}}_{n-1} + 13f^{\mathrm{c}}_{n}),$$

$$y^{\mathrm{p}}_{n+2} = \tfrac{1}{3}(y^{\mathrm{c}}_{n-2} + y^{\mathrm{c}}_{n-1} + y^{\mathrm{c}}_{n}) + \tfrac{1}{12}h(29f^{\mathrm{c}}_{n-2} - 72f^{\mathrm{c}}_{n-1} + 79f^{\mathrm{c}}_{n}),$$

$$y^{\mathrm{c}}_{n+1} = y^{\mathrm{c}}_{n} + \tfrac{1}{12}h(5f^{\mathrm{c}}_{n} + 8f^{\mathrm{p}}_{n+1} - f^{\mathrm{p}}_{n+2}),$$

$$y^{\mathrm{c}}_{n+2} = y^{\mathrm{c}}_{n} + \tfrac{1}{12}h(f^{\mathrm{c}}_{n} + 4f^{\mathrm{p}}_{n+1} + f^{\mathrm{p}}_{n+2}),$$

in which the two predicted steps are computed in parallel, followed by the concurrent evaluation of the two corrections. Note that in this example the solution values are computed at equidistant points (which, in general, need not be the case).

A different approach was adopted by Miranker and Liniger [46] and later refined by Katz, Franklin and Sen [36] in which predictions and higher corrections at past points move forward simultaneously in a diagonal wavefront. An example of this technique is the following method

$$y^{\mathrm{p}}_{n+1} = y^{\mathrm{c}}_{n-1} + 2hf^{\mathrm{p}}_{n},$$

$$y^{\mathrm{c}}_{n} = y^{\mathrm{c}}_{n\ 1} + \tfrac{1}{2}h(f^{\mathrm{p}}_{n} + f^{\mathrm{c}}_{n-1}),$$

in which y^{p}_{n+1} and y^{c}_{n} are computed in parallel.

This latter approach can be generalised so that the computation of the predictor at t_{n+1}, the first corrector at t_n, the second corrector at t_{n-1}, etc., are performed in parallel in conjunction with more sophisticated underlying methods. However, since, in general, relatively few corrections are needed to get suitable behaviour this parallel wavefront approach appears to be of limited value.

The analysis of predictor–corrector methods can be placed in a very general framework as will now be explained.

Consider now an implicit method in which Y_{n+1} is given by the following equation,

$$Y_{n+1} = A_2 \otimes Y_n + hL_1 \otimes F(Y_n) + hL_2 \otimes F(Y_{n+1}), \tag{2.3}$$

where A_2, L_1, and L_2 are square matrices of dimension s. If this equation is solved using a predictor–corrector approach with a Hermite prediction given by

$$Y^{\mathrm{p}} = A_1 \otimes Y_n + hL_0 \otimes F(Y^n) \tag{2.4}$$

and $m-2$ corrections of (2.3), then the ensuing method can be written in a block explicit multivalue format

$$\begin{array}{c|cccccc} I & 0 & & & & & \\ A_1 & L_0 & 0 & & & & \\ A_2 & L_1 & L_2 & 0 & & & \\ \vdots & \vdots & & \ddots & \ddots & & \\ A_2 & L_1 & 0 & \cdots & L_2 & 0 \end{array} \tag{2.5}$$

with sm stages. Specific classes of methods can be constructed on this basis using a Runge–Kutta corrector in (2.3), for example.

Various classes of predictor–corrector methods in the format of (2.5) have been studied:

- Chu and Hamilton [18] constructed a two-block PECE method of order four. However, its stability properties are inferior to those associated with the standard PECE approach based on the Adams methods of order four and five.
- Tam [64] attempted to overcome these drawbacks by constructing classes of methods whose stability regions do not decrease as the order increases. For example, Tam constructs a class of one-corrector methods whose stability region is $\{z\colon |1+z+\frac{1}{2}z^2| \leqslant 1\}$ but unfortunately for methods of order greater than four these methods have very large error coefficients.
- Birta and Abou Rabia [8] used a standard approach and constructed block methods based on equidistant points, while Mouney, Authie and Gayraud [48] implemented an eight-point method of Birta and Abou Rabia on a network of 32 transputers.
- Van der Houwen and Sommeijer [66] constructed classes of methods based on the trivial predictor and a high-order Runge–Kutta corrector and also multivalue block methods with one correction [67].

Jackson and Nørsett [33] and Burrage [12,13] have shown that each application of a corrector increases the order of the overall method by at least one until the order of the corrector is reached. Since block predictor–corrector methods can be placed in the format of a multivalue method the order and stability theories developed for such methods (see Burrage [11] and Hairer and Wanner [29], for example) can be directly applied. Thus Burrage [12,13] has developed a theory which enables the error analysis of any prediction–correction method and has shown that efficient parallel methods are unlikely to be constructed without a rigorous analysis of the local truncation error terms.

In fact numerical testing (see van der Houwen and Sommeijer [66] and Burrage [14]) and theoretical analysis (Skeel and Tam [63]) suggest that the standard approach to prediction–correction, as outlined above, results in methods having relatively small stability regions (unless a large number of corrections are performed) and, in general, large error coefficients because of the complex mixing of predictor and corrector errors. However, a possible way to overcome these difficulties is by the use of splitting techniques which allows more efficient ways of solving (2.3) other than by fixed-point iteration.

Noting that the corrector (2.1) is based on the implicit method,

$$Y = A_2 \otimes Y_n + hL_1 \otimes F(Y_n) + hL_2 \otimes F(Y),$$

a standard technique for solving these nonlinear equations is by some variant of the modified Newton technique and this can be written in its general form as

$$\begin{aligned} Z_n &= A_2 \otimes Y_n + hL_1 \otimes F(Y_n), \\ M_k Y^{(k+1)} &= (M_k - I_s \otimes I_m)Y^{(k)} + Z_n + hL_2 \otimes F(Y^{(k)}), \end{aligned} \tag{2.6}$$

where, for example, $M_k = I_s \otimes I_m$ in the standard fixed-point case while $M_k = I_s \otimes I_m - hL_2 \otimes J_k$, where J_k is some approximation to the Jacobian, in the modified Newton case. It can easily be shown using the iterated tree theory in Jackson and Nørsett [33], Kvaernø [38] or Burrage [14] that if

$$M_k = I_s \otimes I_m + \mathrm{O}(h)$$

then the order of each correction will increase by at least one until the order of the underlying implicit method is reached. Burrage [14] has shown that by choosing the M_k appropriately considerable improvements in efficiency can be gained over the standard prediction–correction approach. For example, Burrage [14] proposes a splitting method based on the Euler predictor and a trapezoidal corrector given by

$$\begin{aligned} y^{(0)} &= y_n + hf(y_n), \\ y^{(k+1)} &= (I - M^{-1})y^{(k)} + M^{-1}\left(y_n + \tfrac{1}{2}hf(y_n)\right) + \tfrac{1}{2}M^{-1}f(y^{(k)}), \end{aligned} \tag{2.7}$$

and analyses different ways of choosing M depending on the nature of the problem. An important class of methods arises if M is diagonal, since in this case the iterations are explicit and can be performed concurrently, but other important cases arise if M is, for example, an incomplete LU factorization. In addition, if the spectrum of the Jacobian of the problem is known, then the iteration described in (2.6) and (2.7) can be made to behave as if it were A-stable although it is an explicit method. This approach to constructing efficient parallel methods based on prediction–correction appears to be a fruitful one although more analysis is needed. However, it would be fair to say that if the standard approach is adopted (fixed-point iteration) then gains in efficiency over extant sequential codes would be meagre at best (see [12,13]) without an analysis of the local error of these methods. Furthermore, if methods are designed with good local error behaviour it seems unlikely that a blocksize greater than 10 would give any additional performance so that speed-ups are limited by this factor.

2.3. Stiff direct methods

Just as it is possible to attempt to construct methods for nonstiff problems based on block explicit Runge–Kutta methods, it is possible to construct methods suitable for stiff problems

based on a block lower triangular Runge–Kutta matrix. Such methods are called "block diagonally implicit Runge–Kutta methods" (BDIRKs). Again the parallelism comes about because the stages within a block are shared amongst the processors. In particular, if the diagonal blocks are themselves diagonal, then such methods are particularly simple to implement. Jackson and Nørsett [32] have shown that if there are k blocks and there are l distinct diagonal coefficients with multiplicity m_i then a bound on the order of the method is

$$1 + \sum_{i=1}^{l} \min(m_i, k).$$

In the case that all the diagonal coefficients are equal this bound is $k + 1$ which is independent of the blocksize and there is little advantage to be gained. However, by allowing at least two distinct diagonal coefficients some small advantages over sequential diagonally implicit methods can accrue (see, for example, a four-stage, two-parallel, two-processor, A_0-stable BDIRK method of order four given in [32]). It is possible to restrict the class of BDIRK methods even further by allowing only a block diagonal structure in which each diagonal block is lower triangular. Lie [42] has constructed an A-stable, four-stage, two-parallel, two-processor method with all diagonal coefficients equal which has order three while Iserles and Nørsett [31] constructed an A-stable method of order four given by

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

where

$$a = \frac{21 + \sqrt{57}}{48}, \qquad b = \frac{9 + 3\sqrt{57}}{16}.$$

However, one of the drawbacks of the above approach stems from the fact that the stage order is at most 2. (The stage order is defined to be the minimum order of all the internal components.) It is now known through the work of Prothero and Robinson [59] and Frank et al. [24] that the order behaviour of a method when applied to many classes of stiff problems is more like the stage order of the method than the classical order of consistency.

One way of overcoming this drawback is to generalize the concept of singly implicitness introduced by Butcher [16] and Burrage [10]. One of the difficulties with implementing implicit Runge–Kutta methods is that in solving the implicit equations for the Y_i a linear system of sm equations has to be solved at each integration step. This comes about when the Y_i in (2.1) are solved by the modified Newton–Raphson method to give

$$\begin{aligned} &(I_s \otimes I_m - hA \otimes J)\Delta = -Y^0 + e \otimes y_n + hA \otimes F(Y^0) \equiv D(Y^0), \\ &Y^1 = Y^0 + \Delta, \end{aligned} \tag{2.8}$$

where each $f_y(t_n + c_j h, Y_j^{(0)})$, $j = 1, \ldots, s$, is approximated by a common $m \times m$ matrix J which can be interpreted to represent the Jacobian of the problem evaluated at a suitable single point.

However, singly implicit methods (SIRKs) have the property that A is similar to a single Jordan block with a real eigenvalue λ of multiplicity s. Thus there exists a nonsingular matrix S and a strictly lower triangular matrix L such that

$$S^{-1}AS = \lambda I_s + L. \tag{2.9}$$

In this case (2.8) can be written as

$$\begin{aligned} &I_s \otimes (I_m - h\lambda J)\Delta = (CS^{-1} \otimes I_m)D(Y^0) - ((C - I_s) \otimes I_m)\Delta, \\ &Y^1 = Y^0 + (S \otimes I_m)\Delta, \\ &C = \lambda(\lambda I_s + L)^{-1}. \end{aligned} \tag{2.10}$$

Usually $C - I_s$ is zero apart from nonzeros on the subdiagonal.

The advantage of this approach is that only one LU factorization of a matrix of dimension m is needed per step but because $C - I_s$ is strictly lower triangular such an implementation is still sequential. In addition, it can be shown that the maximum order of such methods is $s + 1$ (Nørsett and Wolfbrandt [55]) and that a stage order of s is attainable [10].

The implementation as discussed above is not suited to a parallel environment. Consequently Lie [42] and Nørsett and Simonsen [54] suggested a transformation

$$A = T^{-1}DT, \quad D = \operatorname{diag}(\lambda_1, \ldots, \lambda_s).$$

This transformation decouples (2.8) into s systems of equations of the form

$$(I - h\lambda_i J)Z_i = F_i,$$

which can now be solved in parallel independently.

Such methods are called multi-implicit methods. The order and stability properties of multi-implicit methods have been studied by Iserles and Nørsett [31], Bales, Karakashian and Serbin [1], Keeling [37], Orel [58] and Voss and Khaliq [71]. It is known that the maximum order of such methods is $s + 1$ (Nørsett and Wolfbrandt [55]), while Bales et al. [1] have shown that for multi-implicit methods of stage order s all abscissae are real and distinct if $\lambda_j > 0$. Bales et al. [1] and Keeling [37] have shown that all multi-implicit methods of order $s + 1$ are A_0-stable if $\lambda_i \geqslant \frac{1}{2}$, $i = 1, \ldots, s$.

Voss and Khaliq [71] constructed a family of three-stage, L-stable, multi-implicit methods of order three and demonstrated an almost linear speed-up on three processors over the same method implemented on one processor. However, this is not a fair comparison. The comparison should really be made with a three-stage singly implicit method implemented on one processor. If this comparison was made, then it can be shown that multi-implicit methods implemented in parallel would give little improvement over a sequential implementation of a singly implicit method.

Cooper [19] adopted a different approach by attempting a parallel implementation of singly implicit methods of the form

$$\begin{aligned} &I_s \otimes (I_m - h\lambda J)\Delta = (B \otimes I_m)D(Y^0), \\ &Y^1 = Y^0 + \Delta, \end{aligned} \tag{2.11}$$

which allows all the stages to be computed in parallel. The matrix B is chosen to minimize the spectral radius of the amplification matrix. This seems a more viable approach in terms of parallel efficiency than the multi-implicit techniques.

2.4. Stiff indirect methods

In the case that a problem to be solved is stiff a diagonally implicit prediction–correction scheme can be used in which the corrector is split by a positive diagonal matrix. Thus writing the underlying corrector in (2.3) as

$$Y = A_2 \otimes Y_n + hL_1 \otimes F(Y_n) + h(L_2 - D) \otimes F(Y) + hD \otimes F(Y)$$

a fixed-point iteration scheme of the form

$$Y^{(c)} = A_2 \otimes Y_n + hL_1 \otimes F(Y_n) + h(L_2 - D) \otimes F(Y^p) + hD \otimes F(Y^c)$$

can be used. Because D is diagonal these iterations can be performed in parallel with similar implementational costs to sequential DIRKs. Van der Houwen, Sommeijer and Couzy [69] have considered such methods when the underlying corrector method is a Runge–Kutta method. Although in this case the stage order is at most two an analysis by van der Houwen and Sommeijer [68] has shown that it is possible to choose D by minimizing the spectral radius of the amplification matrix over the spectrum of the eigenvalues associated with the problem, which results in a much more improved performance than the limit on the stage order might otherwise have suggested.

3. Parallelism across the system

As has already been mentioned there is no natural parallelism across time when solving IVPs, so that if the system size is moderate then it is necessary to use temporal iterative techniques in order to exploit any massive parallelism. Perhaps one of the simplest techniques in this respect for solving (1.1) is the Picard method. The Picard method generates a sequence of iterative solutions $y^{(0)}(t)$, $y^{(1)}(t), \ldots$ on the region of integration satisfying the equation

$$y'^{(k+1)}(t) = f(t, y^{(k)}(t)), \qquad y^{(k+1)}(t_0) = y_0. \tag{3.1}$$

If the dimension size is m, then this iterative approach allows the decoupling of the problem into m independent parallel quadrature problems. This is a very natural parallelism but unfortunately the iterations converge very slowly unless $\| \partial f / \partial y \|$ is small. Once this decoupling has taken place different quadrature methods are used to discretize the resulting subsystems. In the Picard case parallelism can also be exploited on each individual quadrature problem, so that if N quadrature points are used then the quadrature sums can be formed in O(log N) time on N processors.

For example, consider the application of the Picard method to the standard linear test problem

$$y' = \lambda y, \quad t \in [0, T],$$

then it can be shown that the iterates $y_k(t)$ satisfy the following global error bound

$$|y(t)-y^{(k)}(t)| \leqslant \frac{(\lambda t)^{k+1}}{(k+1)!}, \quad t \in [0, T], \tag{3.2}$$

so that there is no convergence until $k \leqslant \lambda T$. Thus one way to improve the convergence is by the technique of time-windowing in which the region of integration is split up into a series of windows and the iterative process then takes place on each window. Another way to improve the convergence behaviour is via the concept of the splitting of the right-hand side and this leads to the shifted Picard method (see Skeel [62], for example) which takes the form

$$y'^{(k+1)}(t) - M_k y^{(k+1)}(t) = f(t, y^{(k)}(t)) - M_k y^{(k)}(t). \tag{3.3}$$

Clearly there are difficulties in knowing how to choose the window size and the splitting matrix M_k automatically.

The Picard method is a particular example of a much more general class of iterative methods known as waveform relaxation (WR) methods. Waveform relaxation methods were originally introduced by Lelarasmee [40] and Lelarasmee, Ruehli and Sangiovanni-Vincentelli [41] for the time domain analysis of large-scale problems arising from the modelling of integrated circuits. An excellent survey paper in this area was given by White, Sangiovanni-Vincentelli, Odeh and Ruehli [72]. Carlin and Vachoux [17] noted that there can be a slow convergence of the iterations in the case of strong coupling between subsystems. Fortunately for integrated circuit problems this strong coupling occurs only over short time intervals. In addition, the physicality of the problem can be exploited to lump the tightly coupled nodes together, and, in many cases, a reordering of the variables can be made which transforms the system into a mostly triangular form. This arises because of the fact that transistors are highly directional.

A consequence of the above is that waveform relaxation techniques can perform extremely efficiently on problems arising from electrical network modelling. More recently, these techniques have been generalized to general systems of equations with less success because the automatic reordering and efficient groupings of the subsystems is often performed on the basis of physicality which is not always available (see Juang and Gear [35], for example).

Elegant convergence theories (mainly for linear problems) have been developed by Miekkala and Nevanlinna [44,45] and Nevanlinna [49,51] while Bellen, Jackiewicz and Zennaro [3–5] have investigated the stability and contractivity properties of various classes of numerical methods used in a waveform relaxation implementation.

The general form of a continuous-time waveform relaxation method is given by

$$\begin{aligned} &z'^{(k+1)}(t) = G(t, z^{(k+1)}(t), y^{(k+1)}(t), y^{(k)}(t)), \qquad z^{(k+1)}(t_0) = y_0, \\ &y^{(0)}(t_0) = y_0, \\ &y^{(k+1)}(t) = g(t, z^{(k+1)}(t), y^{(k)}(t)), \end{aligned} \tag{3.4}$$

where $G: [t_0, T] \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ and $g: [t_0, T] \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ satisfy

$$G(t, y, y, y) = f(t, y), \qquad g(t, y, y) = f(t, y).$$

The functions G and g are called splitting functions and are chosen in an attempt to decouple (as much as possible) the original system into independent subsystems which can then be solved by a number of differing numerical methods in an independent fashion.

A simpler formulation of (3.4) can be given which is a natural generalization of the Picard approach, namely

$$y'^{(k+1)}(t) = F\big(t,\ y^{(k+1)}(t), y^{(k)}(t)\big), \qquad y^{(k+1)}(t_0) = y_0, \tag{3.5}$$

where the splitting function $F:[t_0, T]\times\mathbb{R}^m\times\mathbb{R}^m\to\mathbb{R}^m$ satisfies

$$F(t,\ y,\ y) = f(t,\ y).$$

There are a number of possibilities for waveform relaxation iteration. Denoting the m components of the function f in (1.1) by $f_1,\ldots,f_m$ and the m components of the kth iterate $y^{(k)}$ by $y_1^{(k)},\ldots,y_m^{(k)}$ and using the simpler formulation given by (3.5), these possibilities include:

- *WR Jacobi*:

$$F_i\big(t,\ y^{(k+1)},\ y^{(k)}\big) = f_i\big(t,\ y_1^{(k)},\ldots,y_{i-1}^{(k)},\ y_i^{(k+1)}, y_{i+1}^{(k)},\ldots,y_m^{(k)}\big), \quad i=1,\ldots,m,$$

  so that (3.5) becomes

$$y_i'^{(k+1)} = f_i\big(t,\ y_1^{(k)},\ldots,y_{i-1}^{(k)},\ y_i^{(k+1)},\ y_{i+1}^{(k)},\ldots,y_m^{(k)}\big), \quad i=1,\ldots,m.$$

- *WR Gauss–Seidel*:

$$F_i\big(t,\ y^{(k+1)},\ y^{(k)}\big) = f_i\big(t,\ y_1^{(k+1)},\ldots,y_i^{(k+1)},\ y_{i+1}^{(k)},\ldots,y_m^{(k)}\big), \quad i=1,\ldots,m.$$

- *WR SOR*:

$$\begin{aligned} &G_i\big(t,\ z^{(k+1)},\ y^{(k+1)},\ y^{(k)}\big)\\ &\quad = f_i\big(t,\ y_1^{(k+1)},\ldots,y_{i-1}^{(k+1)},\ z_i^{(k+1)},\ y_{i+1}^{(k)},\ldots,y_m^{(k)}\big), \quad i=1,\ldots,m,\\ &g_i\big(t,\ z^{(k+1)},\ y^{(k)}\big) = wz_i^{(k+1)} + (1-w)y_i^{(k)}, \quad i=1,\ldots,m. \end{aligned}$$

The advantage of the Jacobi approach is that each component of the system can be solved independently in parallel, while a drawback is that a great deal of past information must be stored if m is large. Storage is not such a problem in the Gauss–Seidel case but then there is no obvious decoupling of the systems. On the other hand it is well known that SOR methods can be parallelized by differing orderings of the components (the so-called chequer board effect). Fang [22] has considered multicoloured implementations on sparse problems.

It is easy to prove an analogous result to the linear case (given in (3.2)) for the convergence of the functions $y^{(k)}(t)$ in the case of the formulation (3.5). Under appropriate differentiability and Lipschitz conditions on F and denoting the maximum norm of a function $g:[a, b]\to\mathbb{R}^m$ by

$$\|g\|_\infty = \max_{t\in[a,b]} |g(t)|,$$

it can be shown that the iterates $y^{(k)}$ converge uniformly to the solution y of (1.1) such that

$$\|y^{(k)}(t) - y(t)\|_\infty \leqslant \frac{C^k(T-t_0)^k}{k!}\|y^{(0)}(t) - y(t)\|_\infty. \tag{3.6}$$

Again it can be seen that if the interval of integration is long, convergence is very slow.

A slightly different approach to the three previous standard iterations is to linearize the problem, as in the Newton waveform approach, in which

$$F\big(t,\ y^{(k+1)},\ y^{(k)}\big) = f\big(t,\ y^{(k)}\big) + f_y\big(t,\ y^{(k)}\big)\big(y^{(k+1)} - y^{(k)}\big)$$

so that

$$y'^{(k+1)} = f(t, y^{(k)}) + f_y(t, y^{(k)})(y^{(k+1)} - y^{(k)}). \tag{3.7}$$

This is now a linear problem and so, for example, quadrature techniques can be used (as in the Picard approach) or techniques based on the fast parallel solution of linear recurrences (see Section 4).

Bellen, Jackiewicz and Zennaro [4,5] have considered the application of continuous Runge–Kutta methods to various waveform relaxation approaches to construct time-point relaxation methods of various types (depending on the underlying waveform). They consider various implementations based on a fixed number of iterations or correcting to convergence and, in addition, investigate order as well as stability behaviour with respect to a number of test equations.

More recently Gear and Xu Xuhai [28] have speculated on a novel approach in which they note that parallelism can easily be exploited in the case of quadrature and nonlinear problems of the form $f(y, t) = 0$, and that these two problems are the limiting case of a differential equation when the stiffness changes from zero to infinity. Thus Gear [27] proposes a blended combination of methods in which the stiff components are solved by a backward differentiation formula and the nonstiff components computed by a quadrature formula, such as those based on Adams methods, in conjunction with a waveform approach. However, numerical testing suggests (as is the case with much of the waveform approach) that this technique is a suitable one only under very special circumstances.

4. Linear IVPs

The situation changes dramatically if the class of problems is restricted to those that are linear. In 1964 Nievergelt [52] proposed a stabbing approach in which the interval of integration is subdivided and a number of different problems are solved on each interval concurrently. Interpolation is then performed in parallel. Nievergelt demonstrated a bounded analysis in the linear case but even in this case as m becomes large the number of solutions grows exponentially. Consequently this approach is now considered not to be practical, especially for nonlinear problems, but it is perhaps the first instance of a specific parallel implementation for a restricted class of problems. In this section we will consider other possibilities.

4.1. The direct approach

Consider the general class of linear problems

$$y'(t) = Q(t)y(t) + g(t), \qquad y(t_0) = y_0 \in \mathbb{R}^m. \tag{4.1}$$

If an s-stage Runge–Kutta method is applied to such a problem over N steps with constant stepsize, then a linear recurrence relationship of the form

$$\begin{aligned} &y_0 = b, \\ &y_{n+1} = R_{n+1}y_n + b_n, \quad n = 0, \ldots, N-1 \end{aligned} \tag{4.2}$$

arises, where the b_n and R_n are related to the vector $(g(t_n + c_1 h)^T, \ldots, g(t_n + c_s h)^T)^T$ and the block diagonal matrix $\text{diag}(Q(t_n + c_1 h), \ldots, Q(t_n + c_s h))$. In the case that the matrix $Q(t)$ is independent of t and $R(z) = p(z)/q(z)$ is the stability function associated with the Runge–Kutta method, then (4.2) can be written as

$$\begin{aligned} &y_0 = b_0, \\ &q(hQ)y_{n+1} = p(hQ)y_n + b_n, \quad n = 0, \ldots, N-1. \end{aligned} \tag{4.3}$$

Thus if any Runge–Kutta method is applied to an arbitrary linear problem a linear recurrence relation of the general form

$$\begin{aligned} &y_0 = b_0, \\ &Q_{i+1} y_{i+1} = P_{i+1} y_i + b_i, \quad i = 0, \ldots, N, \end{aligned} \tag{4.4}$$

is always obtained and this can be written as a block bi-diagonal system ($Cy = b$) of dimension $m(N+1)$ in the form

$$\begin{pmatrix} I & & & \\ P_1 & Q_1 & & \\ & \ddots & \ddots & \\ & & P_N & Q_N \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix}. \tag{4.5}$$

This system of equations can now be solved very efficiently by the technique of cyclic reduction (see Lambiotte and Voight [39] and Heller [30], for example). This technique involves a number of steps. The first step eliminates y_0 from the second equation, y_2 from the fourth, and so on, and subsequent steps involve repeating this process of eliminating couplings between the remaining odd and even rows.

Thus if $m = 1$, the maximum number of processors that can be gainfully employed is $\frac{1}{2}N$ and the system can be solved in O(log N) parallel stages, with the diagonal elements being normalized to unity. In the case that $m > 1$ the situation is more complicated for a number of reasons. First the diagonal blocks should not be normalized but the LU factors formed (this can be done in parallel), secondly the number of processors p available may not be a multiple of m and the cyclic technique may then involve a separator phase.

4.2. The Krylov subspace approach

In the case that $Q(t) \equiv Q$ there is an exact solution to (4.1) namely

$$y(t) = e^{tQ} y_0 + \int_0^t e^{(t-s)Q} g(s)\, ds \tag{4.6}$$

or alternatively

$$y(t+h) = y(t) + \int_0^h e^{(h-s)Q} (y(t+s) + Qy(t))\, ds.$$

Thus the solution can be evaluated if it is possible to compute the exponential of a matrix times a vector v, say, efficiently. This problem has been considered by Gallopoulos and Saad [25] in which they project the problem onto a Krylov subspace K_r of dimension r generated by

$$K_r = \text{span}\{v, Qv, \ldots, Q^{r-1} v\}.$$

The Arnoldi algorithm (the Lanczos algorithm for symmetric matrices) can be implemented in parallel to generate an orthonormal basis W and an upper Hessenberg matrix H, which represents the projection Q onto K_r with respect to this basis such that

$$e^{Qv} \approx \| v \|_2 W e^H e_1, \tag{4.7}$$

where $e_1 = (1, 0, \ldots, 0)^T$. Of course one of the problems here is the appropriate and automatic determination of r. A Runge–Kutta method can now be used to approximate e^H by either a polynomial or rational Padé approximation. Since the problem is likely to be stiff, a real pole Padé approximation can be used (so that the underlying method is multi-implicit). In this case the stability function associated with the method can be written as

$$R(z) = 1 + \sum_{j=1}^{s} \frac{a_j}{z - \lambda_j}$$

so that when approximating $e^H v$ by $R(H)v$ a series of linear systems of the form

$$(H - \lambda_j I_m) x_j = v, \quad j = 1, \ldots, s,$$

is solved in parallel with the solution computed from

$$v + \sum_{j=1}^{s} a_j x_j.$$

4.3. The waveform relaxation approach

Consider again the linear problem

$$y'(t) = Qy(t) + g(t), \qquad y(0) = y_0. \tag{4.8}$$

Suppose that matrix Q is split such that $Q = N - M$, then (4.8) can be written as

$$y'(t) + My(t) = Ny(t) + g(t),$$

and a waveform approach of the form

$$y'^{(k+1)}(t) + My^{(k+1)}(t) = Ny^{(k)}(t) + g(t) \tag{4.9}$$

is a natural one. Furthermore, if M can be made block diagonal, then the system is decomposed into independent subsystems.

Nevanlinna [50] has analysed the convergence properties of the sequence of iterates $y^{(1)}(t)$, $y^{(2)}(t), \ldots$ based on the fact that they can be written in a fixed-point iterative form

$$y^{(k+1)} = Ky^{(k)} + \phi,$$

where K is the convolution operator given by

$$Ku(t) = \int_0^t e^{-(t-s)M} Nu(s)\, ds.$$

Nevanlinna [50] showed that on any bounded interval $[0, T]$,

$$\| K^n \| \leqslant \frac{(CT)^n}{n!}$$

where

$$\|e^{-tM}N\| \leqslant C, \quad \forall t \in [0, T],$$

which again shows that if T is large the convergence is slow.

Nevanlinna [51] investigated the possibility of accelerating this process by solving (4.9) for $z^{(k+1)}(t)$ and then accelerating $y^{(k+1)}(t)$ by

$$y^{(k+1)} = (1 - w_{k+1})z^{(k)} + w_{k+1}z^{(k+1)}.$$

It can be shown that the convergence behaviour of (4.9) depends on the spectral radius of K where

$$\rho(K) = \max_z \rho\big((zI + M)^{-1}N\big) \geqslant \rho(M^{-1}N).$$

O'Leary and White [56] proposed an iterative technique for solving linear systems in which the systems equation matrix is split several times. Jeltsch and Pohl [34] have generalized this work to allow

$$Q = N_i - M_i, \quad i = 1, \ldots, L,$$

and to also allow the overlapping of components which seems to have some improvement on the convergence of the iterates.

Given a waveform approach it is possible to prove various results about the behaviour of numerical methods used in the discretization of the waveform (see Odeh, Ruehli and Carlin [57], for example). Thus in the linear case considered above, if an A-stable linear multistep method is used for the discretization of the waveform, then letting K_h denote the discretization operator it can be shown that

$$\rho(K_h) \leqslant \rho(K),$$

if it is assumed that the whole system is integrated with the same multistep method and with constant stepsize h.

4.4. A multigrid approach

When a system of equations of the form (4.8) is derived from a time-dependent parabolic partial differential equation by a semidiscretization of the spatial variables, a multigrid acceleration of waveform relaxation is possible which exploits the geometry of the problem. This was first analysed by Lubich and Ostermann [43] and Vandewalle and Piessens [70], who extended the approach to nonlinear parabolic partial differential equations. Lubich and Ostermann's approach is independent of the nature of the time discretization.

One of the difficulties with the Jacobi and Gauss–Seidel schemes is the poor convergence behaviour. For example, for the discretized heat equation in which the spatial meshsize is h, these two schemes have convergence rates of $1 - O(h^2)$ (see Miekkala and Nevanlinna [45], for example). A similar behaviour is observed for nonlinear problems. Lubich and Ostermann [43] showed that this convergence behaviour can be improved dramatically if waveform relaxation is combined with the multigrid technique.

Vandewalle and Piessens [70] extended the ideas of Lubich and Ostermann and used a fine-grid smoothing and a coarse-grained correction approach. Thus, assume an initial approximation y_h on a fine grid, then first a number of waveform relaxations are applied to y_h. Secondly, the current approximation is projected onto a coarse grid and the system of equations is solved on this coarse grid with an interpolation of the solution back to the fine grid. Finally a post-smoothing waveform relaxation is applied to the solution.

As Vandewalle and Piessens note any stiff method can be used to solve the differential equations. Nonlinear problems are handled by a Newton waveform approach, while in the linear case they used the Gauss–Seidel approach, both in conjunction with the trapezoidal rule.

5. Conclusions

There are of course other approaches to developing parallel techniques for IVPs which do not necessarily fit into the classifications presented here. For example the stabbing or parallel shooting approach of Nievergelt [52] for linear problems has been generalized by Bellen and Zennaro [7] and Bellen, Vermiglio and Zennaro [6] who noted that any one-step method applied to (1.1) can be viewed as a nonlinear difference equation of the form $y_{n+1} = F(y_n)$. They then proposed an iterative parallel approach (based on Steffensen iterations) to approximate blocks of values $(y_{n+1}, \ldots, y_{n+k})$ concurrently and considered various implementational questions such as stepsize and error control.

Another way in which parallelism can be exploited is in the automatic generation of Taylor series methods. Such work would be an extension of that given by Barton, Willers and Zahar [2]. Currently, very little has been done in terms of parallel implementation (see Bischoff [9]), but this could be very important when, for example, solving large systems of differential algebraic equations.

Finally it should be noted that there are a number of other ways in which parallelism can be exploited without necessarily designing new algorithms. For example, linear combinations of different sequential methods running on different processors can be taken to improve stability and accuracy, while Enright and Higham [21] have used small-scale parallelism in defect evaluation and error control to improve reliability.

In conclusion, it would be fair to say that the subject matter of this review paper is in a state of considerable flux. In the case of the block approach, many authors have hinted and even stated that massive parallelism is possible but a careful analysis of the local error analysis has not been performed in these cases. Burrage [12–14] was the first to do this in a rigorous way and his conclusions are that at best there is only a moderate parallelism available through this approach. This is hardly surprising since a prediction step is for the most part an extrapolatory step so that if the blocksize is too large it takes a large number of corrections to produce an acceptable error.

Waveform relaxation seems a more appropriate vehicle for obtaining large-scale or massive parallelism. But again there are significant problems to be solved before such parallelism can be established. So far only in the case of very structured problems (such as those arising in circuit simulation) or linear problems has waveform relaxation currently shown its worth. Only time will tell but as the saying goes "the jury is still out".

References

[1] L. Bales, O. Karakashian and S. Serbin, On the A_0-acceptability of rational approximations to the exponential function with only real poles, *BIT* 28 (1988) 70–79.

[2] D. Barton, I.M. Willers and R.V.M. Zahar, Taylor series methods for ordinary differential equations—an evaluation, in: I.R. Rice, ed., *Mathematical Software* (Academic Press, New York, 1971) 369–390.

[3] A. Bellen, Z. Jackiewicz and M. Zennaro, Stability analysis of time-point relaxation Heun method, Rept. 1/30, Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo", C.N.R. (1990).

[4] A. Bellen, Z. Jackiewicz and M. Zennaro, Contractivity of waveform relaxation Runge–Kutta methods for dissipative systems in the maximum norm, Rept. 1/61, Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo", C.N.R. (1991).

[5] A. Bellen, Z. Jackiewicz and M. Zennaro, Time point relaxation Runge–Kutta methods for ordinary differential equations, Preprint (1992).

[6] A. Bellen, R. Vermiglio and M. Zennaro, Parallel ODE-solvers with stepsize control, *J. Comput. Appl. Math.* 31 (1990) 277–293.

[7] A. Bellen and M. Zennaro, Parallel algorithms for initial value problems for difference and differential equations, *J. Comput. Appl. Math.* 25 (1989) 341–350.

[8] L.G. Birta and O. Abou'Rabia, Parallel block predictor methods for ODEs, *IEEE Trans. Comput.* 36 (1987) 299–311.

[9] C. Bischoff, Issues in parallel automatic differentiation, Preprint MCS-P235-0491, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL (1991).

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

[11] K. Burrage, Order properties of implicit multivalue methods for ordinary differential equations, *IMA J. Numer. Anal.* 8 (1988) 43–69.

[12] K. Burrage, The error behaviour of a general class of predictor–corrector methods, *Appl. Numer. Math.* 8 (1991) 201–216.

[13] K. Burrage, Efficient block predictor-corrector methods with a small number of corrections, *Appl. Numer. Math.* (submitted).

[14] K. Burrage, The search for the Holy Grail, or: Predictor-corrector methods for solving IVPs, Invited Paper at The International Conference on Parallel Methods for Ordinary Differential Equations, Grado, Italy (1991); also: *Appl. Numer. Math.* 11 (1993) 125–141 (this issue).

[15] K. Burrage and S. Plowman, The numerical solution of ODEIVPs in a transputer environment, in: D.J. Pritchard and C.J. Scott, eds., *Applications of Transputers* 2 (IOS Press, Amsterdam, 1990) 495–505.

[16] J.C. Butcher, On the implementation of implicit Runge–Kutta methods, *BIT* 15 (1976) 358–361.

[17] C.H. Carlin and C. Vachoux, On partitioning for waveform relaxation time-domain analysis of VLSI circuits, in: *Proceedings International Conference on Circuits and Systems*, Montreal, Que. (1984).

[18] M.T. Chu and H. Hamilton, Parallel solution of ODEs by multi-block methods, *SIAM J. Sci. Statist. Comput.* 3 (1987) 342–353.

[19] G.J. Cooper, On the implementation of singly implicit Runge–Kutta methods, *Math. Comp.* (to appear).

[20] J. Demmel., J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling and D. Sorenson, Prospectus for the development of a linear algebra library for high performance computers, Rept. ANL-MCS-TM-97, Argonne National Laboratory, Argonne, IL (1987).

[21] W.H. Enright and D.J. Higham, Parallel defect control, Rept. 237/90. Computer Science Department, University of Toronto, Toronto, Ont. (1990).

[22] M. Fang, A parallel multicoloured Gauss–Seidel solver, Contributed Paper at The International Conference on Parallel Methods for Ordinary Differential Equations, Grado, Italy (1991).

[23] M.F. Flynn, Some computer organizations and their effectiveness, *IEEE Trans. Comput.* 23 (9) (1972) 948–960.

[24] R. Frank, J. Schneid and C.W. Ueberhuber, Order results for implicit Runge–Kutta methods applied to stiff systems, *SIAM J. Numer. Anal.* 22 (1982) 515–534.

[25] E. Gallopoulos and Y. Saad, On the parallel solution of parabolic equations, in: *Proceedings ACM SIGARCH-89* (1989) 17–28.

[26] C.W. Gear, The potential of parallelism in ordinary differential equations, Tech. Rept. UIUC-DCS-R-86-1246, Computer Science Department, University of Illinois at Urbana-Champaign, IL (1986).
[27] C.W. Gear, Parallel solutions of ODEs, in: E.J. Haug and R.C. Deyo, eds., *Real-Time Integration Methods for Mechanical System Simulation*, NATO ASI Series F 69 (Springer, Berlin, 1989) 233–248.
[28] C.W. Gear and Xu Xuhai, Parallelism across time in ODEs, Invited Paper at The International Conference on Parallel Methods for Ordinary Differential Equations, Grado, Italy (1991); also: *Appl. Numer. Math.* 11 (1993) 45–68 (this issue).
[29] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II: *Stiff and Differential-Algebraic Problems* (Springer, Berlin, 1991).
[30] D. Heller, Some aspects of the cyclic reduction algorithm for block tridiagonal systems, *SIAM J. Numer. Anal.* 13 (1978) 484–496.
[31] A. Iserles and S.P. Nørsett, On the theory of parallel Runge-Kutta methods, Tech. Rept. DAMTP NA12/188, Cambridge University, Cambridge, England (1988).
[32] K.R. Jackson and S.P. Nørsett, The potential for parallelism in Runge–Kutta methods, Part 1: RK formulas in standard form, Tech. Rept. 239/90, Computer Science Department, University of Toronto, Toronto, Ont. (1990).
[33] K.R. Jackson and S.P. Nørsett, The potential for parallelism in Runge–Kutta methods, Part 2: RK predictor–corrector formulas (in preparation).
[34] R. Jeltsch and B. Pohl, Waveform relaxation with overlapping systems, Seminar für Angewandte Mathematik, Rept. 91-02, ETH, Zürich, Switzerland (1991).
[35] F. Juang and C.W. Gear, Accuracy increase in waveform Gauss Seidel, Rept. #1518, Department of Computer Science, University of Illinois at Urbana-Champaign, IL (1989).
[36] I.N. Katz, M.A. Franklin and A. Sen, Optimally stable parallel predictors for Adams–Moulton correctors, *Comput. Math. Appl.* 3 (1977) 217–233.
[37] S.L. Keeling, On implicit Runge–Kutta methods with a stability function having distinct real poles, *BIT* 29 (1989) 91–109.
[38] A. Kvaernø, Order of Runge–Kutta methods when using Newton-type iteration, Contributed Paper at The International Conference on Parallel Methods for Ordinary Differential Equations, Grado, Italy (1991).
[39] J. Lambiotte and R. Voight, The solution of tridiagonal linear systems on the CDC-star-100, *ACM Trans. Math. Software* 1 (1974) 308–329.
[40] E. Lelarasmee, The waveform relaxation method for the time domain analysis of large scale nonlinear dynamical systems, Ph.D. Thesis, University of California, Berkeley, CA (1982).
[41] E. Lelarasmee, A. Ruehli and A. Sangiovanni-Vincentelli, The waveform relaxation method for time domain analysis of large scaled integrated circuits, *IEEE Trans. CAD IC Syst.* 1 (1982) 131–145.
[42] I. Lie, Some aspects of parallel Runge–Kutta methods, Mathematics and Computation Rept. 3/87, NIT, Trondheim, Norway (1987).
[43] C. Lubich and A. Ostermann, Multi-grid dynamic iteration for parabolic equations, *BIT* 27 (1987) 216–234.
[44] U. Miekkala and O. Nevanlinna, Convergence of dynamic iterations for initial value problems, *SIAM J. Sci. Statist. Comput.* 8 (1987) 459–482.
[45] U. Miekkala and O. Nevanlinna, Sets of convergence and stability regions, *BIT* 27 (1987) 557–584.
[46] W.L. Miranker and W. Liniger, Parallel methods for the numerical integration of ordinary differential equations, *Math. Comp.* 21 (1967) 303–320.
[47] F.R. Moulton, *New Methods in Exterior Ballistics* (University of Chicago Press, Chicago, IL, 1926).
[48] G. Mouney, G. Authie and T. Gayraud, Parallel solution of ODEs, Implementation of block methods on transputer networks, Contributed Paper at ICIAM91, Washington, DC (1991).
[49] O. Nevanlinna, Remarks on Picard–Lindelöf iteration, Part I, *BIT* 29 (1989) 328–346.
[50] O. Nevanlinna, Remarks on Picard–Lindelöf iteration, Part II, *BIT* 29 (1989) 535–562.
[51] O. Nevanlinna, Linear acceleration of Picard–Lindelöf iteration, *Numer. Math.* 57 (1990) 147–156.
[52] J. Nievergelt, Parallel methods for integrating ordinary differential equations, *Comm. ACM* 7 (2) (1964) 731–733.
[53] S.P. Nørsett, Experiments with parallel ODE solvers, Invited Paper at the International Conference on Parallel Methods for Ordinary Differential Equations, Grado, Italy (1991).

[54] S.P. Nørsett and H.H. Simonsen, Aspects of parallel Runge–Kutta methods, in: A. Bellen, C.W. Gear and E. Russo, eds., *Numerical Methods for Ordinary Differential Equations, Proceedings Aquila Symposium,* Lecture Notes in Mathematics 1386 (Springer, Berlin, 1989) 103–107.

[55] S.P. Nørsett and A. Wolfbrandt, Attainable order of rational approximations to the exponential function with only real poles, *BIT* 17 (1977) 200–208.

[56] D.P. O'Leary and R.E. White, Multi-splittings of matrices and parallel solution of linear systems, *SIAM J. Algebraic Discrete Methods* 6 (1986) 630–640.

[57] F. Odeh, A.F. Ruehli and C.H. Carlin, Robustness aspects of an adaptive waveform relaxation scheme, in: *Proceedings International Conference on Computer Design*, Port Chester, NY (1983).

[58] B. Orel, Real pole approximations to the exponential function, Tech. Rept. 1/90, Math. Sciences, NIT, Trondheim, Norway (1989).

[59] A. Prothero and A. Robinson, On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations, *Math. Comp.* 28 (1974) 145–162.

[60] J.B. Rosser, A Runge–Kutta for all seasons, *SIAM Rev.* 9 (1967) 417–452.

[61] L.F. Shampine and H.A. Watts, Block implicit one-step methods, *Math. Comp.* 23 (1969) 731–740.

[62] R.D. Skeel, Waveform iteration and the shifted Picard splitting, *SIAM J. Sci. Statist. Comput.* 10 (1989) 756–776.

[63] R.D. Skeel and H.W. Tam, Potential for parallelism in explicit linear methods, Computer Science Department, University of Illinois at Urbana-Champaign, IL (1991).

[64] H.W. Tam, Parallel methods for the numerical solution of ordinary differential equations, Ph.D. Thesis, Computer Science Department, University of Illinois at Urbana-Champaign, IL (1989).

[65] L.G. Valiant, A scheme for fast parallel communication, *SIAM J. Comput.* 11 (2) (1982) 350–361.

[66] P.J. van der Houwen and B.P. Sommeijer, Variable step iteration of high order Runge–Kutta methods on parallel computers, Tech. Rept. NM-R8817, CWI, Amsterdam (1988).

[67] P.J. van der Houwen and B.P. Sommeijer, Block Runge–Kutta methods on parallel computers, Tech. Rept. NM-R8906, CWI, Amsterdam (1989).

[68] P.J. van der Houwen and B.P. Sommeijer, Iterated Runge–Kutta methods on parallel computers, CWI, Amsterdam (1991); also: *SIAM J. Sci. Statist. Comput.* 12 (5) (1991) 1000–1028.

[69] P.J. van der Houwen, B.P. Sommeijer and W. Couzy, Embedded diagonally implicit Runge–Kutta algorithms on parallel computers, Tech. Rept. NM-R8912, CWI, Amsterdam (1989); also: *Math. Comp.* 58 (1992) 135–159.

[70] S. Vandewalle and R. Piessens, Numerical experiments with nonlinear multigrid waveform relaxation on a parallel processor, *Appl. Numer. Math.* 8 (1991) 149–161.

[71] D.A. Voss and A.Q.M. Khaliq, L-stable parallel methods for parabolic problems, Rept., Department of Mathematics, Western Illinois University, Macomb, IL (1989).

[72] J. White, A. Sangiovanni-Vincentelli, F. Odeh and A. Ruehli, Waveform relaxation: theory and practice, *Trans. Soc. Comput. Simulation* 2 (1985) 95–133.

[73] Z. Zlatev, Treatment of some mathematical models describing long-range transport of air pollutants on vector processors, *Parallel Comput.* 6 (1988) 87–98.

[74] Z. Zlatev, Computations with large and band matrices on vector processors, in: H. van der Vorst and P. van Dooren, eds., *Parallel Algorithms for Numerical Linear Algebra*, Advances in Parallel Computing 1 (Elsevier Science Publishers, Amsterdam, 1990) 7–37.

[75] Z. Zlatev and R. Beckowicz, Numerical treatment of large-scale air pollution models, *Comput. Math. Appl.* 16 (1988) 93–109.