A Survey of Parallel Numerical Methods for Initial Value Problems for Ordinary Differential Equations

Kenneth R. Jackson[1]
Computer Science Department, University of Toronto,
Toronto, Ontario, Canada M5S 1A4.
<krj@na.toronto.edu>

Abstract

The parallel solution of Initial Value Problems for Ordinary Differential Equations has become an active area of research during the past few years. We briefly survey the recent developments in this area, with particular emphasis on traditional forward-step methods that offer the potential for effective small-scale parallelism on currently existing machines.

Introduction

It is widely believed that the only feasible means of solving many important computationally intensive problems in science and engineering is to use parallel computers effectively. As a result, increasing numbers of researchers have begun investigating numerical methods for a wide variety of advanced machine architectures. In this paper, we briefly survey parallel numerical methods for Initial Value Problems (IVPs) for Ordinary Differential Equations (ODEs):

$$\begin{aligned} y'(x) &= f(x, y(x)), \quad \text{for } x \in [x_0, x_e], \\ y(x_0) &= y_0, \end{aligned} \tag{1}$$

where $y : \mathbb{R} \rightarrow \mathbb{R}^m$ and $f : \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}^m$. See also the earlier reviews of Gear [15, 16] and Burrage [7]. Although the development of parallel algorithms in this area has lagged that in several other fields, such as linear algebra and partial differential equations, activity in parallel methods for IVPs has recently increased significantly. Most of this research, though, is preliminary in nature, its goal being the exploration of new parallel algorithms rather than the implementation, testing and evaluation of efficient, reliable, robust mathematical software.

In the next section, we begin with a general discussion of the need for parallelism in IVP solvers as well as sources of and impediments to parallelism. In the third section, we consider the potential for *small-scale* parallelism in traditional forward-step methods, with particular emphasis on Runge-Kutta schemes. These methods are suitable for machines with a few processors (e.g., 2–10) and fast inter-processor communication, two properties characteristic of shared-memory multiprocessors. We end with a list of references for methods having the potential for both small- and large-scale parallelism. A more extensive, up-to-date bibliography on parallel methods for both Initial- and Boundary-Value Problems for ODEs may be obtained upon request from the author.

We originally intended to review as well novel IVP methods offering the potential for *large-scale* parallelism. These include *dynamic iteration* schemes — often called *waveform-relaxation* methods — as well as techniques based upon the fast solution of linear recurrence relations. However, space constraints prevented us from adequately surveying in this brief article methods for both small- and large-scale parallelism, two distinctly different classes of schemes. An important consideration in our choice to review the more traditional forward-step methods was that some of these schemes are currently sufficiently well understood to be implemented effectively on existing shared-memory multiprocessors. On the other hand, most of the novel methods, offering the potential for large-scale parallelism, require further investigation before effective methods can be implemented. Moreover, shared-memory machines with a few processors, appropriate for the former class of methods, are currently widely available and relatively easy to use. This is not the case for massively parallel systems needed to run the novel methods. Although we believe the novel IVP methods will play an important role in scientific computing in the future, IVP methods having the potential for small-scale parallelism are likely to be of more value to the practitioner now. The reader interested in large-scale parallelism for IVPs should see [1, 3, 4, 17, 18, 29, 30, 31, 33, 37, 38, 41, 43, 44, 46, 47, 48, 51, 54, 60] and the references therein. For the applications of these methods to partial differential equations, see [13, 14, 35, 36].

General Discussion

The desire for parallel IVP solvers arises from the need to solve many important problems more rapidly than is currently possible. This may be because the solution is needed in real time, as is the case for flight simulators or control systems, or it may be because computation time on a conventional sequential machine is so large that it adversely affects the productivity of engineers and scientists working on the design of complex systems, for example. There is also the technological imperative that *chips are cheap*,

[1] This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario

giving manufacturers the ability to build inexpensive parallel machines with massive computational potential. The challenge to computational scientists is to exploit this potential to solve problems more efficiently and/or reliably than they could in the past, or to solve problems that were previously intractable.

As discussed more fully in [53], the IVP (1) can be time consuming to solve if

- f is expensive to evaluate, as might be the case if each f evaluation requires the solution of an auxiliary problem,
- the number of equations, m, in the system is large, a property characteristic of spatially-discretized PDEs or large electrical circuits,
- the interval of integration $[x_0, x_e]$ is long, or
- the IVP must be solved repeatedly, as happens in parameter fitting problems.

Gear [15, 16] classifies the means of achieving parallelism in IVP solvers into two main categories:

- *parallelism across the method* or equivalently *parallelism across time*, and
- *parallelism across the system* or equivalently *parallelism across space*,

with the second name for each category being motivated by the parallel solution of time-dependent PDEs. Included in the first class are algorithms that exploit several concurrent function evaluations within each step, as do the Block and Runge-Kutta methods discussed in the next section, as well as techniques that solve for many steps simultaneously, as do the fast parallel methods for linear recurrence relations [17, 52] and some dynamic iteration schemes. Other dynamic iteration schemes, exploiting *modular integration* for example, fall into the second class of methods, as do several more obvious techniques such as exploiting parallelism in the evaluation of f as well as in performing the vector operations and in solving the linear and nonlinear equations that arise at each step of an otherwise standard IVP solver. Furthermore, we emphasize that many of these techniques are complementary: a method might, for example, compute several f evaluations simultaneously while assigning several processors to compute each f. Although exploiting parallelism in the f evaluations and linear and nonlinear algebra within an IVP method can be effective [6, 42, 61, 62], we do not discuss this approach further, since the focus of this paper is on the IVP methods themselves.

There are several impediments to parallelism, the seemingly natural forward propagation of information in IVPs being a prime example. However, this is not always as severe an impediment as one might first think. Gear [17] notes that the problems $y' = f(x)$, $y(x_0) = y_0$, and $0 = f(x, y)$ — in some sense two extreme limiting cases of the IVP (1) — are "embarrassingly" parallel. He also observes that fast methods for the solution of linear IVPs can be constructed from fast algorithms for the parallel solution of linear recurrence relations [17, 52]. Further study is needed to determine the character and extent of this impediment.

Two other impediments to parallelism, common to most application areas, are

- the *narrowness of the computation lattice*, an extreme case of which is a portion of the computation that must be performed on one processor, and
- the need for synchronization, forcing the computation to halt on one processor while waiting for information from another,

both of which are discussed by Gear [15] in the context of IVP solvers. In discussing *Amdahl's law*, Gustavson [19] notes that the narrowness of the computation graph is often not as serious an impediment to parallelism as it first seems, since, as the problem size grows, the time to execute the narrow sections frequently remains constant, hence requiring a decreasing portion of the total execution time. The need for synchronization is a considerably more severe constraint, particularly for the traditional forward-step methods discussed in the next section.

Because computation versus communications rates vary dramatically across parallel machine architectures, it is essential to match the needs of a method to the capabilities of a machine. This suggests that a useful way to divide parallel methods is between those that require fast communication between processors to function efficiently and those for which this is not as important a consideration. In the next section, we consider traditional forward-step methods which require machines with a few processors only but rapid communication between them, two properties characteristic of shared-memory multiprocessors. The dynamic iteration methods, not reviewed in this paper, can effectively use many more processors with slower inter-processor communication speeds, characteristic of message-passing machines.

Small-Scale Parallelism in Traditional IVP Methods

Two of the earliest papers on parallel methods for IVPs were by Miranker and Liniger [39] and Miranker [40]. They point out that many standard predictor-corrector schemes based on multistep formulas, such as

$$\begin{aligned} y^p_{n+1} &= y^c_n + \frac{h}{2}\left(3f(x_n, y^c_n) - f(x_{n-1}, y^c_{n-1})\right) \quad (2) \\ y^c_{n+1} &= y^c_n + \frac{h}{2}\left(f(x_{n+1}, y^p_{n+1}) + f(x_n, y^c_n)\right), \quad (3) \end{aligned}$$

are inherently sequential: y^c_n must be computed before y^p_{n+1}, which in turn must be computed before y^c_{n+1}, etc. However, if we substitute the predictor

$$y^p_{n+1} = y^c_{n-1} + 2hf(x_n, y^p_n), \quad (4)$$

for (2), then both y^p_{n+1} and y^c_n from (4) and (3), respectively, can be computed simultaneously. Setting

$$Y_n = \begin{pmatrix} y^p_{n+1} \\ y^c_n \end{pmatrix} \quad \text{and} \quad F_n = \begin{pmatrix} f(x_{n+1}, y^p_{n+1}) \\ f(x_n, y^c_n) \end{pmatrix},$$

we can rewrite the predictor-corrector pair (4)–(3) as an explicit one-step *block method*:

$$Y_n = AY_{n-1} + hBF_{n-1} \tag{5}$$

where

$$A = \begin{bmatrix} 0 & I \\ 0 & I \end{bmatrix}, \qquad B = \begin{bmatrix} 2I & 0 \\ \frac{1}{2}I & \frac{1}{2}I \end{bmatrix}.$$

Miranker and Liniger [39] go on to develop a theory for more general multistep block methods of this type and propose several schemes suitable for machines with a few processors (e.g., 2 to 4).

More recently, several other authors [5, 7, 8, 11, 12, 32, 56, 57, 58, 23, 59] have considered similar explicit k-block r-value schemes of the form

$$Y_n = \sum_{i=1}^{k} A_i Y_{n-i} + h \sum_{i=1}^{k} B_i F_{n-i}, \tag{6}$$

where $Y_n \in \mathbb{R}^{rm}$ consists of r y-values and $F_n \in \mathbb{R}^{rm}$ is f evaluated at those y-values. (Formula (5) is a 1-block 2-value scheme of this type.) These methods have the characteristic that all r f evaluations within each F_n can be performed simultaneously. Hence, these methods, like all others considered in this section, are best suited for machines with a few processors and fast inter-processor communication, two properties characteristic of shared-memory multiprocessors.

The example above suggests a simple general paradigm for achieving parallelism across time in traditional forward-step methods: group the stages of a method into blocks for which all function evaluations associated with each block can be performed simultaneously.

Simonsen [53] exploits the natural parallelism of this type inherent in extrapolation schemes to construct an effective parallel method. The paradigm can also be applied to Runge-Kutta (RK) methods. Miranker and Liniger [39] derived a class of parallel RK (PaRK) methods, but their schemes are ineffective because of poor stability. We briefly review below other classes of PaRK methods, both as instructive examples of variants of the general paradigm noted above and also because PaRK schemes are a promising class of methods. Many of the examples can be easily generalized to a broader class of methods, but we felt it best to illustrate them for RK schemes, since these are better understood by a wider audience.

An s-stage RK formula may be written as

$$Y_{n,i} = y_n + h \sum_{j=1}^{s} a_{ij} F_{n,j}, \quad i = 1, \ldots, s \tag{7}$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j F_{n,j} \tag{8}$$

where the $\{Y_{n,i}\}$ are the *internal stage values*, $\{F_{n,i} = f(x_n + c_i h, Y_{n,i})\}$ the associated function values, and the $\{a_{ij}\}$, $\{b_j\}$, $\{c_j\}$ are the coefficients of the formula. The latter are frequently displayed in tableau form as

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

where $A = [a_{ij}]$ is a $s \times s$ matrix, $b = (b_i)$ and $c = (c_i)$ are s-vectors.

The general paradigm enunciated above can be applied directly to both *Explicit* RK (ERK) formulas (for which $a_{ij} = 0$ for $i \le j$) and *Diagonally-Implicit* RK (DIRK) formulas (for which $a_{ij} = 0$ for $i < j$). We seek formulas for which the coefficient matrix A can be written in block form as

$$A = \begin{bmatrix} D_1 & 0 & 0 & \cdots & 0 \\ A_{21} & D_2 & 0 & \cdots & 0 \\ A_{31} & A_{32} & D_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \\ A_{p1} & A_{p2} & \cdots & A_{p\,p-1} & D_p \end{bmatrix} \tag{9}$$

where, for an ERK formula, each D_k is the zero matrix and, for a DIRK formula, each D_k is a (possibly different) diagonal matrix. Thus, for any $k \in \{1, \ldots, p\}$, all $\{Y_{n,i}\}$ in the k^{th} block of an ERK formula can be computed in parallel once all $\{F_{n,j}\}$ in blocks $1, \ldots, k-1$ are available. After the $\{Y_{n,i}\}$ have been computed, all $\{F_{n,i}\}$ associated with the k^{th} block can be evaluated simultaneously in preparation to compute the $k+1^{\text{st}}$ block of $\{Y_{n,i}\}$. Similarly, for any $k \in \{1, \ldots, p\}$, each $Y_{n,i}$ in the k^{th} block of a DIRK formula depends on itself through $F_{n,i}$ and the previously computed $\{F_{n,j}\}$ in blocks $1, \ldots, k-1$ only. Thus, all $Y_{n,i}$ and the associated $F_{n,i}$ in the k^{th} block can be computed simultaneously by solving independent systems of m equations each. Consequently, the ERK and DIRK variants, respectively, retain their characteristic explicit or diagonally-implicit property.

Regrettably, these ERK schemes offer little potential for parallelism [21, 27], as the order of a p-block ERK formula cannot exceed p and, if the order is p, the stability region is $\{z : |\sum_{i=0}^{p} z^i/i!| \le 1\}$, which is not large. The construction of parallel ERK formulas and some minor advantages of these schemes are discussed in [21, 27]. However, a predictor-corrector variant discussed below seems more promising.

On the other hand, parallel DIRK formulas offer some advantage. For example, Iserles and Nørsett [26] derive a family of 4-stage, 4^{th}-order 2-parallel DIRK formulas, which includes the L-stable (but not B-stable) formula

$$\begin{array}{c|cccc} 1/2 & 1/2 & 0 & 0 & 0 \\ 2/3 & 0 & 2/3 & 0 & 0 \\ 1/2 & -5/2 & 5/2 & 1/2 & 0 \\ 1/3 & -5/3 & 4/3 & 0 & 2/3 \\ \hline & -1 & 3/2 & -1 & 3/2 \end{array}$$

for which $Y_{n,1}$ and $Y_{n,2}$ can be computed simultaneously, after which $Y_{n,3}$ and $Y_{n,4}$ can be computed simultaneously.

Moreover, because $a_{11} = a_{33}$ and $a_{22} = a_{44}$, only two matrices need to be factored to solve for all four $Y_{n,i}$ by Newton's method. Other formulas of this type are given in [22, 26, 27, 34], where restrictions on the attainable order and stability for this class of formulas are considered.

It is instructive to write (7) in the tensor product form

$$Y_n = e_s \otimes y_n + h(A \otimes I_m) F_n \tag{10}$$

where e_s is the vector of s ones, I_m is the $m \times m$ identity matrix, $Y_n = (Y_{n,i})_{i=1}^{s} \in \mathbb{R}^{sm}$, $F_n = (F_{n,i})_{i=1}^{s} \in \mathbb{R}^{sm}$, and

$$\Gamma \otimes \Lambda = \begin{bmatrix} \gamma_{11}\Lambda & \gamma_{12}\Lambda & \cdots & \gamma_{1l}\Lambda \\ \vdots & \vdots & & \vdots \\ \gamma_{k1}\Lambda & \gamma_{k2}\Lambda & \cdots & \gamma_{kl}\Lambda \end{bmatrix} \in \mathbb{R}^{k\mu \times l\nu},$$

for any matrices $\Gamma = [\gamma_{ij}] \in \mathbb{R}^{k \times l}$ and $\Lambda \in \mathbb{R}^{\mu \times \nu}$. Consider applying either simple iteration

$$Y_n^{(k+1)} = e_s \otimes y_n + h(A \otimes I_m) F_n^{(k)} \tag{11}$$

or a variant of Newton's method

$$N_n^k \, (Y_n^{(k+1)} - Y_n^k) = e_m \otimes y_n + h(A \otimes I_m) F_n^{(k)} - Y_n^{(k)} \tag{12}$$

to solve (10), where $Y_n^{(k)}$ is the k^{th} approximation to Y_n, $F_n^{(k)}$ is F_n evaluated at $Y_n^{(k)}$ instead of Y_n, and N_n^k is an approximation to the Newton iteration matrix. We see immediately that the application of either (11) or (12) to solve (10) leads to a block formulation of the method in which all s f evaluations needed for $F_n^{(k)}$ at each iteration can be performed simultaneously. This specific observation concerning the inherent parallelism in (11) and (12) generalizes to the application of iterative methods in many contexts.

Butcher [9] proposed an effective scheme for solving the linear equations associated with (12). His technique transforms this system of sm coupled linear equations to s linear systems of m equations. If A can be diagonalized, then these s systems are independent and can therefore be solved simultaneously. Several authors [2, 27, 24, 25, 50] have studied these transformations for PaRK schemes, with particular emphasis on the restrictions that this technique places on the choice of coefficients for *Fully-Implicit* RK formulas. This work has yielded some promising theoretical and numerical results for the solution of stiff IVPs.

The inherent parallelism in both (11) and (12) can be exploited in *Predictor-Corrector* (PC) variants of RK formulas. In this context, we introduce an extension of (11) proposed in [22]:

$$Y_n^{(k+1)} = e_s \otimes y_n + h[(A - D) \otimes I_m] F_n^{(k)} + h(D \otimes I_m) F_n^{(k+1)} \tag{13}$$

where D is a diagonal matrix. Since $Y_n^{(k+1)}$ is an argument of $F_n^{(k+1)}$, we must solve for $Y_n^{(k+1)}$ in (13). However, each $Y_{n,i}^{(k+1)}$ is independent of all other $Y_{n,j}^{(k+1)}$, giving (13) a diagonally-implicit character. Moreover, it follows that (13) is composed of s independent systems of m equations each, and these can therefore be solved simultaneously.

If the initial guess $Y_n^{(0)}$ for Y_n depends only on y_n and possibly $Y_n^{(0)}$ itself, then the iterations (11) and (13) yield a "true" RK formula with coefficient matrix of the form (9); (12) yields a Rosenbrock method. Promising results for RK PC methods based on (11) and (13) are reported in [20, 21, 22, 28]. If, on the other hand, the initial guess $Y_n^{(0)}$ depends on previously computed values other than y_n, then the resulting PC formula looses the one-step property of the underlying RK formula. Although this adds to the complexity of the scheme, results in [7, 8, 28] suggest that this approach may yield very effective parallel methods.

Several authors have studied the potential for parallelism in *General Linear Methods* (GLMs). (GLMs [10] form a broad class of methods including all the schemes considered above.) Tam [56] and Skeel and Tam [55] analyze the stability regions of explicit GLMs, and show that in some sense parallelism cannot improve the stability of explicit methods. However, Tam's results [56, 57, 58] show that explicit parallel methods afford one more opportunity than sequential methods to obtain a formula that has both a large stability region and small truncation errors. Burrage's analysis [8] of truncation errors in PC GLMs contributes to our understanding of the derivation of such schemes. However, much more work in this area is needed.

References

[1] S. Aslam and C. W. Gear, "Asynchronous integration of ordinary differential equations on multiprocessors", Tech. Rep. UIUCDCS–R–89–1525, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1988.

[2] L. Bales, O. Karakashian and S. Serbin, "On the A_o-acceptability of rational approximations to the exponential function with only real poles", *BIT*, vol. 28, pp. 70–79, 1988.

[3] A. Bellen, "Parallelism across the steps for difference and differential equations", in *Numerical Methods for Ordinary Differential Equations*, A. Bellen, C. W. Gear and E. Russo (eds.), Proceedings, L'Aquila, 1987, Lecture Notes in Mathematics #1386, Springer, Berlin, 1989, pp. 22–35.

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

[5] L. G. Birta and O. Abou-Rabia, "Parallel block predictor-corrector methods for ode's", *IEEE Trans. Comput.*, vol. C–36, pp. 299–311, 1987.

[6] A. Bose, I. Nelken and J. Gelfand, "A comparison of several methods of integrating stiff ordinary differential equations on parallel computing architectures", in *Proc. of the Third Hypercube Conf.*, Pasadena, CA, pp. 1712–1716, 1988.

[7] K. Burrage, "Solving nonstiff IVPs in a transputer environment", manuscript, CMSR, Univ. of Liverpool,

Liverpool, England, 1989.

[8] K. Burrage, "The error behaviour of a general class of predictor-corrector methods", manuscript, CMSR, University of Liverpool, Liverpool, England, 1989.

[9] J. C. Butcher, "On the implementation of implicit Runge-Kutta methods", *BIT*, vol. 16, pp. 237–240, 1976.

[10] J. C. Butcher, "General linear methods: a survey", *Appl. Numer. Math.*, vol. 1, pp. 273–284, 1985.

[11] M. T. Chu and H. Hamilton, "Parallel solution of ODEs by multi-block methods", *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. 342–353, 1987.

[12] M. A. Franklin, "Parallel solution of ordinary differential equations", *IEEE Trans. Comput.*, vol. C–27, pp. 413–420, 1978.

[13] E. Gallopoulos and Y. Saad, "On the parallel solution of parabolic equations", in *Proc. 1989 ACM Int'l Conf. on Supercomputing*, Herakleion, Greece, pp. 17–28, 1989.

[14] E. Gallopoulos and Y. Saad, "Efficient solution of parabolic equations by polynomial approximation methods", Tech. Rep. 969, CSRD, Univ. of Illinois, Urbana, IL, 1990.

[15] C. W. Gear, "The potential for parallelism in ordinary differential equations", Tech. Rep. UIUCDCS–R–86–1246, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1986.

[16] C. W. Gear, "The parallel methods for ordinary differential equations", Tech. Rep. UIUCDCS–R–87–1369, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1987.

[17] C. W. Gear, "Massive parallelism across the method in ODEs", Tech. Rep. UIUCDCS–R–88–1442, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1988.

[18] C. W. Gear and D. Wang, "Explicit stiff stability via splitting and the parallel solution of ODEs", Tech. Rep. UIUCDCS–R–87–1328, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1987.

[19] J. L. Gustavson, "Reevaluating Amdahl's law", *Comm. ACM*, vol. 31, pp. 532–533, 1988.

[20] P. J. van der Houwen and B. P. Sommeijer, "Variable step iteration of high-order Runge-Kutta methods on parallel computers", Tech. Rep. NM–R8817, Dept. of Numer. Math., Centre for Math. and Comp. Sci., Amsterdam, The Netherlands, 1988.

[21] P. J. van der Houwen and B. P. Sommeijer, "Parallel iteration of high-order Runge-Kutta methods with stepsize control", *J. Comput. Appl. Math.*, vol. 29, pp. 111–127, 1990.

[22] P. J. van der Houwen, B. P. Sommeijer and W. Couzy, "Embedded diagonally implicit Runge-Kutta algorithms on parallel computers", Tech. Rep. NM–R8912, Dept. of Numer. Math., Centre for Math. and Comp. Sci., Amsterdam, The Netherlands, 1989.

[23] P. J. van der Houwen, B. P. Sommeijer and P. A. van Mourik, "Note on explicit parallel multistep Runge-Kutta methods", *J. Comput. Appl. Math.*, vol. 27, pp. 411–420, 1989.

[24] O. A. Karakashian and W. Rust, "On the parallel implementation of implicit Runge-Kutta methods", *SIAM J. Sci. Stat. Comput.*, vol. 9, pp. 1985–1090, 1988.

[25] S. L. Keeling, "On implicit Runge-Kutta methods with a stability function having distinct real poles", *BIT*, vol. 29, pp. 91–109, 1989.

[26] A. Iserles and S. P. Nørsett, "On the theory of parallel Runge-Kutta methods", to appear in IMA J. Numer. Anal., 1990.

[27] K. R. Jackson and S. P. Nørsett "The potential for parallelism in Runge-Kutta methods. Part 1: RK formulas in standard form", Tech. Rep. No. 239/90 Comp. Sci. Dept., Univ. of Toronto, Toronto, Canada, 1990; submitted to *Math. Comp.*

[28] K. R. Jackson and S. P. Nørsett "The potential for parallelism in Runge-Kutta methods. Part 2: RK predictor-corrector formulas", in preparation to be submitted to *Math. Comp.*

[29] F.-L. Juang, "Accuracy increase in waveform relaxation", Tech. Rep. UIUCDCS–R–88–1466, Comp. Sci. Dept., Univ. of Illinois at Urbana-Champaign, Urbana, IL, 61801, 1988.

[30] F.-L. Juang, "Waveform methods for ordinary differential equations", Tech. Rep. UIUCDCS–R–90–1563, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1990.

[31] F.-L. Juang and C. W. Gear, "Accuracy increase in waveform Gauss Seidel", Tech. Rep. UIUCDCS–R–89–1518, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1988.

[32] I. N. Katz, M. A. Franklin and A. Sen, "Optimally stable parallel predictors for Adams-Moulton correctors", *Comp. and Maths. with Appls.*, vol. 3, pp. 217–233, 1977.

[33] E. Lelarasmee, A. Ruehli and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits", *IEEE Trans. Computer-Aided Design*, vol. CAD–1, pp. 131–145, 1982.

[34] I. Lie, "Some aspects of parallel Runge-Kutta methods", Math. and Comp. Rep. 3/87, Numer. Math. Dept., Norwegian Inst. of Tech., Trondheim, Norway, 1987.

[35] I. Lie and R. Skålin, "Relaxation-based integration by Runge-Kutta methods and its application to moving finite element methods", manuscript, NDRE, Electronics Div., P. O. Box 25, N–2007, Kjeller, Norway, 1989.

[36] Ch. Lubich and A. Ostermann, "Multi-grid dynamic iteration for parabolic equations", *BIT*, vol. 27, pp. 216–234, 1987.

[37] U. Miekkala and O. Nevanlinna, "Convergence of dynamic iteration methods for initial value problems", *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. 459–482, 1987.

[38] U. Miekkala and O. Nevanlinna, "Sets of convergence and stability regions", *BIT*, vol. 27, pp. 554–584, 1987.

[39] W. L. Miranker and W. Liniger, "Parallel methods for the numerical integration of ordinary differential equations", *Math. Comp.*, vol. 21, pp. 303–320, 1967.

[40] W. L. Miranker, "A survey of parallelism in numerical analysis", *SIAM Review*, vol. 13, pp. 524–547, 1971.

[41] D. Mitra, "Asynchronous relaxation for the numerical solution of differential equations by parallel computer", *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. s43–s56, 1987.

[42] I. Nelken, T. S. Ho, H. Rabitz and J. Gelfand, "Parallel methods in sensitivity analysis", in *Proc. of the Fourth Hypercube Conf.*, Monterey, CA, 1989.

[43] O. Nevanlinna, "Remarks on Picard-Lindelöf iteration: Part I", *BIT*, vol. 29, pp. 328–346, 1989.

[44] O. Nevanlinna, "Remarks on Picard-Lindelöf iteration: Part II", *BIT*, vol. 29, pp. 535–562, 1989.

[45] O. Nevanlinna, "A Note on Picard-Lindelöf iteration", in *Numerical Methods for Ordinary Differential Equations*, A. Bellen, C. W. Gear and E. Russo (eds.), Proceedings, L'Aquila, 1987, Lecture Notes in Mathematics #1386, Springer, Berlin, 1989, pp. 97–102.

[46] O. Nevanlinna and F. Odeh, "Remarks on the convergence of waveform relaxation method", *Numer. Funct. Anal. and Optimiz.*, vol. 9, pp. 435–445, 1987.

[47] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation", *IEEE Trans. on Computer-Aided Design*, vol. CAD–3, pp. 308–330, 1984.

[48] J. Nievergelt, "Parallel methods for integrating ordinary differential equations", *Comm. ACM*, vol. 7, pp. 731–733, 1964.

[49] S. P. Nørsett and H. H. Simonsen, "Aspects of parallel Runge-Kutta methods", in *Numerical Methods for Ordinary Differential Equations*, A. Bellen, C. W. Gear and E. Russo (eds.), Proceedings, L'Aquila, 1987, Lecture Notes in Mathematics #1386, Springer, Berlin, 1989, pp. 103–117.

[50] B. Orel, *Real pole approximations to the exponential function*, Tech. Rep. 1/90, Math. Sci. Div., Norwegian Inst. of Tech., Trondheim, Norway, 1989; to appear in *BIT*.

[51] J. Sand and S. Skelboe, "Stability of backward Euler multirate methods and the convergence of waveform relaxation", submitted to *BIT*, 1990.

[52] U. Schendel, *Introduction to Numerical Methods for Parallel Computers*, New York: Ellis Horwood, 1984.

[53] H. H. Simonsen, "Extrapolation methods for ODE's: continuous approximations, a parallel approach", Ph. D. thesis, Math. Sci. Div., Norwegian Inst. of Tech., Trondheim, Norway, 1990.

[54] R. D. Skeel, "Waveform iteration and the shifted Picard splitting", *SIAM J. Sci. Stat. Comput.*, vol. 10, pp. 756–776, 1989.

[55] R. D. Skeel and H. W. Tam, "Potential for parallelism in explicit linear methods", manuscript, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1989.

[56] H. W. Tam, "Parallel methods for the numerical solution of ordinary differential equations", Ph. D. thesis, Tech. Rep. UIUCDCS–R–89–1516, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1989.

[57] H. W. Tam, "One-stage parallel methods for the numerical solution of ordinary differential equations", submitted to *SIAM J. Sci. Stat. Comput.*, 1989.

[58] H. W. Tam, "Two-stage parallel methods for the numerical solution of ordinary differential equations", submitted to *SIAM J. Sci. Stat. Comput.*, 1989.

[59] P. B. Worland, "Parallel methods for the numerical solution of ordinary differential equations", *IEEE Trans. Comput.*, vol. C–25, pp. 1045–1048, 1976.

[60] Xu Xuhai and C. W. Gear, "Potential performance of methods for parallelism across time in ODEs", Tech. Rep. UIUCDCS–R–90–1587, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1990.

[61] G-C. Yang, "Paraspice: a parallel circuit simulator for shared-memory multiprocessors", in *27th ACM/IEEE Design Automation Conf.*, pp. 400–405, 1990.

[62] G-C. Yang, "Paraspice: a parallel direct circuit simulator for shared-memory multiprocessors", Ph. D. thesis, Comp. Sci. Dept., Univ. of Illinois, Urbana, IL, 1990.