P.J. van der Houwen
Centrum voor Wiskunde en Informatica
Amsterdam, The Nederlands

PARALLEL ITERATION SCHEMES FOR IMPLICIT ODEIVP METHODS

Abstract

In this contribution to the Proceedings of the "International Symposium on New Aspects of Numerical Analysis in the Light of Recent Technology" held at Stresa on 13 until 18 September 1993, we give a survey of recent research at CWI for solving the implicit relations arising in ODEIVP methods on parallel computers. Starting with a General Linear method as introduced by Butcher, three forms of parallelism for solving the associated implicit equations are discussed, viz. (i) parallelism across the stages within a single step (stage parallelism), (ii) parallel preconditioners, and (iii) parallelism across the steps (step parallelism). The structure of these types of parallel methods will be described.

1 Introduction

Consider the initial value problem for ordinary differential equations

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{f}(\mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}, \mathbf{f} \in \Re^d, \quad t_0 \leq t \leq T. \qquad (1)$$

The greater part of the numerical integration methods for solving this problem (ODEIVP methods) can be presented in the form of a k–dimensional General Linear Method (GL method) as introduced by Butcher [6] (see also [7]):

$$\mathbf{Y}_n = (E \otimes I_d)\mathbf{Y}_{n-1} + h(B \otimes I_d)\mathbf{F}(\mathbf{Y}_n), \quad n = 1, \ldots, N. \qquad (2)$$

Here, $\mathbf{Y}_n$ is a block vector with k vectorial components (stages) $\mathbf{Y}_{n,i}$, $i = 1, 2, \ldots, k$, of dimension d which are assumed to present numerical approximations to the exact solution $\mathbf{y}(t)$ at the k points $t_{n-1} + c_j h$, $i = 1, 2, \ldots, k$, where h denotes the stepsize and the c_j define the k–dimensional abscissa vector $\mathbf{c}$. It will also be assumed that $c_k = 1$ and we define the step points t_n by $t_n := t_{n-1} + h$. The block vector $\mathbf{F}(\mathbf{Y}_n)$ contains the k derivatives $\mathbf{f}(\mathbf{Y}_{n,i})$ and the k–by–k matrices E and B contain the method parameters. Finally, the matrix I_d is the d–by–d identity matrix, $\otimes$ denotes the Kronecker product, and $\mathbf{Y}_0$ contains the starting values for the numerical method.

The GL method (2) defines in each step a system of kd equations. Not all equations are necessarily implicit. Following Butcher (1987, p.367), we shall assume that there are rd explicit and sd implicit

equations with $r + s = k$ where $s \leq k$. For example, Runge–Kutta (RK) methods fit into (2) with

$$E := \begin{pmatrix} 0 & \dots & 0 & 1 \\ \vdots & \dots & \vdots & \vdots \\ \vdots & \dots & \vdots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} A & \mathbf{0} \\ \mathsf{b}^T & 0 \end{pmatrix}, \tag{3}$$

where A and b present the familiar arrays appearing in the Butcher tebleau representation of RK methods. Thus, RK methods have $r = 1$ and $s = k - 1$. It will be assumed that the equations are arranged such that the implicit equations correspond to the first s rows of the matrices E and B

On sequential computers, one usually selects explicit GL methods ($s = 0$) for *nonstiff* problems and implicit GL methods, for *stiff* problems ($s > 0$). If $s = 0$, than B is strictly lower triangular, so that the GL method (1) requires $N(k - 1)$ righthand side evaluations. If $s > 0$, then each step requires the solution of a system of sd equations. These equation are usually solved by means of Newton iteration. In order to reduce the computational costs, the matrix B is chosen to be *lower triangular* where the first s diagonal entries b_{ii} (corresponding to the implicit equations) equal some constant nonzero value. In analogy with the terminology used in RK methods, we shall call such GL methods *diagonally implicit GL methods.* This leads to a system of s equations, each of dimension d with the same Jacobian matrix. Taking this as a first indication of the computational cost involved, we conclude that the computational costs

for solving stiff problems on sequential problems on sequential computers by diagonally implicit GL methods are $O(N_{LU}d^3)$ arithmetic operations, where N_{LU} denotes the number of updates of the Jacobian matrix of the implicit equations (for general matrices B, this would be as much as $O(N_{LU}s^3d^3)$ operations). Of course, both for stiff and nonstiff problems, the above restrictions on the matrix B have consequences for the accuracy and stability of the corresponding GL method.

On parallel computers, it is possible to iterate implicit GL methods with arbitrary matrix B and with an arbitrary number of stages without increasing the *sequential* cost of the method. Here, sequential costs means that all righthand side evaluations, LU decompositions, etc., that can be done in parallel are counted as just one righthand side evaluation, one LU decomposition, etc. Thus, parallel computers enables us to use the most accurate and stable GL methods possible, without restricting the matrices E and B to a special form. So far, the construction of such GL methods did not receive much attention and, at present, the classical RK methods of Butcher–Kuntzmann and the Radau IIA methods seem to offer the best starting point for parallel iteration (the definition of these classical RK methods can be found in Butcher [6]).

This paper will survey various parallel iteration techniques for implicit ODEIVP methods (of the GL form (2)), that have been investigated by the numerical group at CWI. Weshall distinguish

(i) parallelism across the stages within a single step (stage parallelism)

(ii) parallel preconditioners

(iii) parallelism across the steps (step parallelism).

The structure of the iteration schemes will be discussed in the following subsections. For numerical experiments we refer to the CWI publications listed among the references.

2 Stage parallelism

We approximate the solution $\mathbf{Y}_n$ of (2) by successive $\mathbf{Y}_n^{(j)}$, $j = 1, 2, \ldots$, such that $\mathbf{Y}_n^{(j)} \to \mathbf{Y}_n$ as $j \to \infty$. The iteration scheme relating the iterates consists of a *predictor formula* providing $\mathbf{Y}_n^{(1)}$ and a *correction formula* providing the subsequent iterates $\mathbf{Y}_n^{(j)}$. The method (2) itself will be referred to as the generating *corrector.*

The most simple iteration scheme first iterate for $n = 1$ to obtain $\mathbf{Y}_1$, then it iterates for $n = 2$ to obtain $\mathbf{Y}_2$, etc. Thus, representing the iterates $\mathbf{Y}_n^{(j)}$ by points in the (n, j)–plane, the order of computation is (necessarily) *column wise.*

We shall restrict out considerations to iteration schemes defined by the formula pair

$$\mathbf{Y}_n^{(1)} - h\,(D^* \otimes I_d)\,\mathbf{F}\,(\mathbf{Y}_n(1)) =$$

$$(E^* \otimes I_d)\,\mathbf{Y}_{n-1} + h\,(B^* \otimes I_d)\,\mathbf{F}\,(G^*\mathbf{Y}_{n-1})\,, \tag{4}$$

$$\begin{aligned}
&\mathbf{Y}_n^{(j)} - h\,(D \otimes I_d)\,\mathbf{F}\,(\mathbf{Y}_n(j)) = \\
&\quad (E \otimes I_d)\,\mathbf{Y}_{n-1} + h\,((B-D) \otimes I_d)\,\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right), \\
&\quad j = 2, \ldots, m
\end{aligned}$$

where $n = 1, 2, \ldots, N$. D and D^* are diagonal matrices of which the last r diagonal entries are zero, and where the set of matrices $\{D^*, E^*, B^*, G^*\}$ define the predictor formula. Possible options are predictors based on the last step value, and on extrapolation or backward differentiation of stage values from the preceding step. Those predictors will be referred to as the LSV predictor, the EXP predictor, and the BDF predictor, respectively, and can be defined by matrix sets $\{D, E, B-D, E\}$, $\{D, E, B-D, E^*\}$, and $\{D, E^*, \emptyset, \emptyset\}$, where E^* is a free matrix to control accuracy and stability. Notice that the predictor and correction formula are equally expansive. This feature will be useful when we adapt the iteration scheme to achieve step parallelism (see Section 4). Evidently, if the iterates $\mathbf{Y}_n^{(j)}$ satisfying (4) converge to fixed vectors $\mathbf{V}_n$ as $j \to \infty$, then $\mathbf{V}_n = \mathbf{Y}_n$. The integer m is assumed to be sufficiently large, so that numerically $\mathbf{Y}_n^{(m)} = \mathbf{Y}_n$.

For nonstiff problems, it is allowed to set $D = D^* = \emptyset$, by which the iteration scheme reduces to fixed point iteration. The corresponding integration method will be called a *PIGL method* (Parallel Iterated GL method). For stiff problems, it is crucial that the first s

diagonal entries of D assume suitably chosen positive values (if B^* does not vanish, this also applies to D^*). The resulting method will be referring to as *PDIGL methods* (Parallel Diagonally Iterated GL methods).

When using predictor formulas the (sequential) costs of which equal those of one correction iteration (like the LSV, EXP, and BDF predictor defined above), the total cost of the P(D)IGL method consists of Nm iterations. Each iteration of the P(D)IGL method possesses parallelism across the stages, because all derivative components of the block vectors $\mathbf{F}(G^*\mathbf{Y}_{n-1})$ and $\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right)$ can be computed in parallel, resulting in m sequential righthand sides per step. In the case of nonvanishing D^* and D, we also have to solve a system of the s equations of dimension d in each iteration. However, by virtue of the diagonal structure of the matrices D^* and D, these s equations are uncopuled, so that they can be solved in parallel. In particular, the expansive LU decompositions corresponding to the s equations can be obtained concurrently, resulting in $O(N_{LU}d^3)$ sequential arithmetic operations, where N_{LU} denotes the number of updates of the Jacobian matrix of the implicit equation that needs the most updates. The sequential costs of the PIGL method and the explicit GL method and in the diagonally implicit method mentioned in Section 1 are comparable. Likewise, the sequential costs of the PDIGL method and the diagonally implicit method GL method of Section 1 are comparable if the number N_{LU} of LU updates are comparable.

In the case where the GL method (2) is an RK method, the iteration scheme (4) has been extensively studied. For $D = D^* = \emptyset$, convergence and stability results and performance evaluation can be found in Lie [23], Nørsett & Simonsen [26], Jackson & Nørsett [20, 21], Burrage [3, 4, 5], Jackson, Kværnø& Nørsett [22], and in Van der Houwen & Sommerijer [11]. The methods arising for non-vanishing D and D^* have been investigates in [12, 13, 14, 27].
We present a few results for the scalar test equation

$$\frac{dy(y)}{dt} = \lambda y(t), \tag{5}$$

where λ runs through the spectrum of the Jacobian matrix $\partial \mathbf{f}/\partial \mathbf{y}$.

Theorem 2.1 *With respect to the test equation (5) the correction formula (4) is convergent if*

$$\rho\left(Z(z)\right) < 1, \quad Z(z) := z\left(I - zD\right)^{-1}\left(B - D\right), \quad z := \lambda h,$$

where $\rho\left(Z\right)$ denotes the spectral radius of the iteration matrix Z. □

The region in the complex z–plane where the convergence condition is satisfied will be called *convergence region.* If the convergence region contains the whole lefthand plane, the iteration scheme will be called *A–convergent.* Evidently, if $D = \emptyset$, then the convergence region is given by the disk

$$|z| < \frac{1}{\rho\left(B\right)}, \tag{6}$$

so that we only have A–convergence if $\rho(B)$ vanishes.

Example 2.1: Let the generating corrector (2) be the s–stage RK method of Butcher–Kuntzmann and let $D = \emptyset$ in (4). Then the radius of the convergence region of (4) is given by

$s=1$	$s=2$	$s=3$	$s=4$	$s=5$
2.00	3.48	4.54	5.88	7.14

If $D \neq \emptyset$, then a necessary condition for A–convergence is that the spectral radius of $Z(z)$ is less that 1 at infinity, i.e.,

$$\rho(Z(\infty)) = \rho\left(D^{-1}B - I\right) < 1 \tag{7}$$

This observation suggest choosing D such that $\rho(Z(\infty))$ is minimized. In [12, 27] it was shown that for the 2–stage, 3–stage, 4–stage Radau IIA corrector, this approach does lead to A–convergent iteration schemes.

Theorem 2.2 *With respect to the rest equation (5), the stability region of the P(D)IGL method is the intersection of the convergence region of the correction formula in (4) and the stability region of the corrector (2).* □

3 Parallel preconditioners

Let us define the residual function associated with the corrector (2):

$$\mathbf{R}_n(\mathbf{Y}) := \mathbf{Y} - (E \otimes I_d)\mathbf{Y}_{n-1} - h(B \otimes I_d)\mathbf{F}(\mathbf{Y}) . \qquad (8)$$

Then the correction formula in the P(D)IGL method (4) can be written in the form

$$\mathbf{Y}_n^{(j)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j)}\right) =$$
$$\mathbf{Y}_n^{(j-1)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right) - \mathbf{R}_n\left(\mathbf{Y}_n^{(j-1)}\right), j = 2, \ldots, m$$

The convergence of the iteration scheme can be accelerated by introducing a preconditioning matrix P in front of the residual function:

$$\mathbf{Y}_n^{(j)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j)}\right) =$$
$$\mathbf{Y}_n^{(j-1)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right) - P\mathbf{R}_n\left(\mathbf{Y}_n^{(j-1)}\right), j = 2, \ldots, m \qquad (9)$$

The choice of the preconditioner can be based on the following analogue of theorem 2.1:

Theorem 3.1 *With respect to the test equation (5) the correction formula in (9) is convergent if*

$$\rho(Z(z)) < 1, \quad Z(z) := z(I - zD)^{-1}(I - P + zPB - zD), \quad z := \lambda h,$$

where $\rho(Z)$ denotes the spectral radius of the iteration matrix Z. □

For RK corrector, the constructions of preconditioners yielding suitable iteration matrices $Z(z)$ has been investigated in [10, 15, 16]. One of the main results derived in these papers immediately carries over to the case of GL corrector.

Theorem 3.2 *Let J denote the Jacobian matrix of the righthand side function of the IVP and define*

$$P := (I - hD \otimes J)^{-1}(I - 2hD \otimes J + hB \otimes J). \qquad (10)$$

Then, with respect to the test equation (5), the iteration matrix is given by

$$Z(z) = z^2(I - zD)^{-2}(D^2 - 2DB + B^2).$$

□

We shall call the method defined by the correction formula (9) with preconditioner (10) a preconditioned P(D)IGL method. This method requires the evaluation (or update) of the Jacobian J. However, since the computational work involved can be done in parallel with the other computational tasks, the sequential costs are not increased (note that for $D \neq \emptyset$, the LU decomposition of $I - hD \otimes J$ needed for applying P is already available).
Evidently, if $D = \emptyset$, then the convergence condition of the preconditioned P(D)IGL method is identical with (6). However, because of the factor z^2 in the iteration matrix, the rate of convergence is much

better. If $D \neq \emptyset$, we are again led to consider the iteration matrix at infinity leading to a necessary condition for A–convergence:

$$\rho(Z(\infty)) = \rho(I - 2D^{-1}B + D^{-2}B^2) < 1. \qquad (11)$$

As before, this suggest choosing D such that $\rho(Z(\infty))$ is minimized. In Van der Houwen & Sommaijer [16] it was shown that for the 2–stage, 3–stage and 4–stage Radau IIA corrector, this approach does lead to A–convergent iteration schemes.

Theorem 3.3 *With respect to the test equation (5), the stability region of the preconditioned P(D)IGL method is the intersection of the convergence region of the correction formula { (9), (10) } and the stability region of the corrector (2).* □

4 Step parallelism

In the preceding sections, the first iterate $\mathbf{Y}_n^{(1)}$ of the nth step is only computed if the iterates $\mathbf{Y}_{n-1}^{(j)}$ corresponding to the $(n-1)$st step have converged to $\mathbf{Y}_{n-1}$. Hence, the solutions $\mathbf{Y}_n$ of the corrector (2) are computed *sequentially*, that is, the iterates $\mathbf{Y}_n^{(j)}$, when represented by points in the (n, j)–plane, are computed necessarily *column–wise*, so that there is no parallelism across the steps. In this section, we consider iteration schemes that allows simultaneous iteration at a number of step point resulting in *step–parallel methods.*

4.1 Jacobi–type correction formula

The most simple approach, and at the same time the most effective with regard to parallelism, in getting step–parallel methods computes the iterates $\mathbf{Y}_n^{(j)}$ *row–wise*. Correction formulas allowing row–wise orderings have been investigated by the Trieste group. Steffenson correction formulas were analyzed in Bellen et al. [1, 2], and an extension to Newton–type iteration in Chartier [8]. Let us consider the related Jacobi–type correction formula

$$\begin{aligned} &\mathbf{Y}_n^{(j)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j)}\right) = \\ &(E \otimes I_d)\mathbf{Y}_{n-1}^{(j-1)} + h((B-d) \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right) , \end{aligned} \tag{12}$$

where for each $j = 2, \ldots, m$, the time index n runs from 1 until N. It is easily seen that this formula allows row–wise computation of the iterates. The total sequential cost of (12) consists of the cost needed to compute the sequence $\{\mathbf{Y}_n^{(1)} : n = 1, \ldots, N\}$ and the cost of $m-1$ corrector. Hence, if the costs of computing the sequence $\{\mathbf{Y}_n^{(1)} : n = 1, \ldots, N\}$ can be ignored, for example, by using formulas like $\mathbf{Y}_n^{(1)} = \mathbf{y}_0 \otimes \mathbf{e}$, $\mathbf{e}$ being the vector unit entries, then the total sequential costs are about the number of iterations m. This seems to be considerably less than the total costs of the P(D)IGL of the preceding sections which required Nm iterations. However, the drawback of this approach is the need of rather accurate first iterates $\{\mathbf{Y}_n^{(1)} : n = 1, \ldots, N\}$ and the poor convergence factors associated with (12). If the initial iterates are not sufficiently accurate, and that seems to be likely when their computational costs are to be negligible, than the iteration process easily diverges and

if it does converge, then the number of iterations may be extremely large.

4.2 Gauss–Seided–type correction formula

With respect to convergence speed, the conventional PC correction formula in (4) and the Jacobi–type correction formula (12) are extreme cases. Again referring to the representation of the iterates $\mathbf{Y}_n^{(j)}$ by points in the (n,j)–plane, we see that in both cases the correction formula for $\mathbf{Y}_n^{(j)}$ needs a "lefthand neighbor" and a "lower neighbor". However, the accuracy of these "neighbouring" iterates differs greatly. In the conventional PC correction formula the accuracy is the best possible, whereas in the Jacobi–type correction formula, the accuracy is worst. Therefore, we now consider an "intermediate" ordering in which the iterates are computed *diagonal–wise* leading to the Gauss–Seidel–type correction formula

$$\begin{aligned} &\mathbf{Y}_n^{(j)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j)}\right) = \\ &(E \otimes I_d)\mathbf{Y}_{n-1}^{(j)} + h((B-d) \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right) , \end{aligned} \tag{13}$$

where $j = 2, \ldots, m$ and $n = 1, \ldots, N$. It is easily seen that all iterates with with $j + n =$ constant can be computed concurrently. Assuming that the predictor formula is equally expansive as one correction, we conclude that the diagonal ordering requires $N + m$ sequential iterations. The advantage is that the accuracy of the "lefthand neighbor" and "lower neighbor" is much better than in the Jacobi–type correction formula (12), but at the cost of less

massive parallelism. Diagonal computation of iterates has already been used by Miranker and Liniger [24] where the iterates produced by Adams-type PECE methods were computed in parallel along diagonals.

4.3 Dynamic Gauss–Seidel–type correction formula

Still, the Gauss–Seidel correction formula (13) may also fail in practice. A remedy is offered by the *dynamic* Gauss–Seidel correction formula

$$\begin{aligned} &\mathbf{Y}_n^{(j)} - h(D \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j)}\right) = \\ &\qquad (E \otimes I_d)\mathbf{Y}_{n-1}^{(q(n-1,j))} + h((B-D) \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right), \qquad (14) \\ &q(n,j) := j + j^*(t_n) - 1, \quad j = 2,\ldots,m; \quad n = 1,\ldots,N \end{aligned}$$

where the value of $j^*(t_n)$ is determined dynamically during the integration process. For example, by using predictor formulas of the form

$$\begin{aligned} &\mathbf{Y}_n^{(j)} - h(D^* \otimes I_d)\mathbf{F}\left(\mathbf{Y}_n^{(1)}\right) = \\ &(E^* \otimes I_d)\mathbf{Y}_{n-1}^{(q(n-1,1))} + h(B^* \otimes I_d)\mathbf{F}\left(G^*\mathbf{Y}_{n-1}^{(q(n-1,1))}\right), \end{aligned} \qquad (15)$$

and by the condition that the iterate $\mathbf{Y}_{n-1}^{(q(n-1,1))}$ is sufficiently accurate to obtain a reliable first iterate $\mathbf{Y}_n^{(1)}$.

We remark that the other correction formulas discussed in this paper can also be represented in the form (14) by an appropriate definition of the ordering function $q(n,j)$. Ignoring the Jacobi–type

Correction formula	$q(n,j)$	Predictor	Seq. iterations
P(D)IGL (cf. (4))	m	$B^* \neq \emptyset$	Nm
Gauss-Seidel (cf. (13))	j	$B^* \neq \emptyset$	$N+m-1$
Dynamic Gauss-Seidel	$j+j^*(n)-1$	$B^* \neq \emptyset$	$\sum_n j^*(t_n)+m$

Table 1: Number of sequential iterations associated with {(14), (15)}

correction formula (12) which is too unreliable, Table 1 lists these q-functions together with the sequential costs associated with the predictor–correction formula pair.

For RK–based correction formulas, a convergence analysis of step–parallel methods described above can be found in [17, 18].

References

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

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

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

[4] K. Burrage *The search for the Holy Grail, or Predictor Corrector methods for solving ODEIVPs*, Appl. Numer. Math. 11 (1993), pp. 125–141.

[5] K. Burrage *Efficient block predictor–corrector methods with small number of iterations*, J. Comput. Appl. Math. 45 (1993), pp. 139–150.

[6] J.C. Butcher *On the convergence of numerical solution to ordinary differential equations*, Math. Comp. 20, pp. 1–10.

[7] J.C. Butcher *The numerical analysis of ordinary differential equations, Runge–Kutta and general linear methods*, Wiley 1987, New York.

[8] P. Chartier *Parallelism in the numerical solution of the initial value problems for ODEs and DAEs*, Thesis (1993), Universit de Rennes I, France.

[9] E. Hairer, G. Winner *Solving ordinary differential equations, II. Stiff and differential-algebraic problems*, Springer–Verlag 1991, Berlin.

[10] J.P. van der Houwen *Preconditioning in implicit initial value problem methods on parallel computers*, Advances in Computational Mathematics 1 (1993), pp. 39–60.

[11] J.P. van der Houwen, B.P. Sommeijer *Parallel iterations of high-order Runge-Kutta methods with stepsize control*, J. Comput. Appl. Math. 29 (1990), pp. 111-127

[12] J.P. van der Houwen, B.P. Sommeijer *Iterate Runge-Kutta methods on parallel computers*, SIAM J. Sci. Stat. Comput. 12 (1991), pp. 1000-1028.

[13] J.P. van der Houwen, B.P. Sommeijer, W. Couzy *Embedded diagonally implicit Runge-Kutta algorithms on parallel computers*, Math. Comp. 58 (1992), pp. 135–139.

[14] J.P. van der Houwen, B.P. Sommeijer *Analysis of parallel diagonal implicit iteration of Runge-Kutta methods*, Appl. Numer. Math. 11 (1992) pp. 169–188.

[15] J.P. van der Houwen, B.P. Sommeijer *Butcher-Kuntzmann methods for nonstiff problems on parallel computers*, to appear in Numerical Algorithms.

[16] J.P. van der Houwen, B.P. Sommeijer *Preconditioning in parallel Runge-Kutta methods for nonstiff initial value problems*, to appear in Comp. Math. Applic.

[17] J.P. van der Houwen, B.P. Sommeijer, W.A. van der Veen *Parallel iterations across the steps of high order Runge-Kutta methods for nonstiff initial value problems*, submitted for publication.

[18] J.P. van der Houwen, B.P. Sommeijer, W.A. van der Veen *Parallelism Across the Steps in Iterated Runge–Kutta Methods for Stiff Initial Value Problems*, submitted for publication.

[19] A. Iserles, S.P. Nørsett *On the theory of parallel Runge–Kutta methods*, IMA J. Numer. Anal. 10 (1990), pp. 463–448.

[20] K.R. Jackson, S.P. Nørsett *Parallel Runge-Kutta methods* (manuscript).

[21] K.R. Jackson, S.P. Nørsett *The potential for parallelism in Runge–Kutta methods, Part I: RK formulas in standard form*, Technical Report No. 239/90 (1990), Department of Computer Science, University of Toronto.

[22] K.R Jackson, A. Kværnø, S.P. Nørsett *Order of Runge–Kutta methods when using Newton–type iteration*, Technical Report No. 1/92 (1992), Division of Math. Sciences, University of Trondheim.

[23] I. Lie *Some aspects of parallel Runge–Kutta methods*, Report 3/87 (1987), Dept. of Mathematics, University of Trondheim.

[24] W.L. Miranker, W. Liniger *Parallel methods for the numerical integration of ordinary differential equation*, Math. Comp. 21 (1967), pp. 303–320.

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

[26] S.P. Nørsett, H.H. Simonsen *Aspects of parallel Runge–Kutta methods*, in A. Ballen, C.W. Gear, E. Russo (Eds): Numerical Methods for Ordinary Differential Equations, Proceedings L'Aquila 1987, LMN 1386, Springer–Verlag, Berlin.

[27] B.P. Sommeijer *Parallelism in the numerical integration of initial value problems*, Thesis defended at the University of Amsterdam (1992).