Parallel iterative linear solvers for multistep Runge-Kutta methods

E. Messina, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9619 1996

Report NM-R9619
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Parallel Iterative Linear Solvers for Multistep Runge–Kutta Methods

E. Messina [1], J. J. B. de Swart [2] and W. A. van der Veen [2]

[1] *Dipartimento di Matematica e Applicazioni "R. Caccippoli",*
University of Naples "Federico II",
Via Cintia, I-80126 Naples, Italy,

[2] *CWI, P.O. Box 94079, 1090 GB Amsterdam, the Netherlands*

Abstract

This paper deals with solving stiff systems of differential equations by implicit Multistep Runge–Kutta (MRK) methods. For this type of methods, nonlinear systems of dimension sd arise, where s is the number of Runge–Kutta stages and d the dimension of the problem. Applying a Newton process leads to linear systems of the same dimension, which can be very expensive to solve in practice. Like in [HS96], where the one-step RK methods were considered, we approximate these linear systems by s systems of dimension d, which can be solved in parallel on a computer with s processors. In terms of Jacobian evaluations and LU-decompositions, the k-step s-stage MRK applied with PILSMRK on s processors is equally expensive as the widely used k-step Backward Differentiation Formula on 1 processor, whereas the stability properties are better than that of BDF. If both methods perform the same number of Newton iterations, then the accuracy delivered by the new method is also higher than that of BDF.

AMS Subject Classification (1991): Primary: 65L05. Secondary: 65F05, 65F50.
CR Subject Classification (1991): G.1.7, G.4
Keywords & Phrases: numerical analysis, Newton iteration, Multistep Runge–Kutta methods, parallelism.
Note: The research reported in this paper was partly supported by STW (Dutch Foundation for Technical Sciences).

1. Introduction

For solving the stiff initial value problem (IVP)

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \quad y, f \in \mathbf{R}^d, \quad t_0 \le t \le t_e, \tag{1.1}$$

a widely used class of methods is that of the Backward Differentiation Formulae (BDFs)

$$y_n = (\kappa^T \otimes I) y^{(n-1)} + h_n \beta f(y_n).$$

Here, $\otimes$ denotes the Kronecker product and the vector $y^{(n-1)}$ is defined by $(y_{n-k}^T, \ldots, y_{n-1}^T)^T$, where y_j^T approximates the solution at $t = t_j$ and k is the number of previous steppoints that are used for the computation of the approximation in the current time interval. The stepsize $t_{n+1} - t_n$ is denoted by h_n. The scalar β and the k-dimensional vector κ contain the method parameters. They depend on $h^{(n)}$, which is the vector with k previous stepsizes defined by $h^{(n)} := (h_{n-k+1}, \dots, h_n)^T$. In the sequel, I stands for the identity matrix and e_i for unit vector in the ith direction. The dimensions of I and e_i may vary, but will always be clear from the context.

For example, the popular codes DASSL [Pet91] and VODE [BHB92] are based on BDFs. However, a drawback of BDFs is the loss of stability if the number of steppoints k increases. As a consequence of Dahlquist's order barrier, no A-stable BDF can exceed order 2. Moreover, BDFs are not zero-stable for $k > 6$.

A promising class of methods that can overcome these drawbacks of BDFs are the Multistep Runge–Kutta (MRK) methods, which are of the form

$$y_n = (\chi^T \otimes I)y^{(n-1)} + h_n(\alpha^T \otimes I)F(Y_n), \tag{1.2}$$

where Y_n is the solution of the equation

$$R(Y_n) = 0, \quad R(Y_n) := Y_n - (G \otimes I)y^{(n-1)} - h_n(A \otimes I)F(Y_n). \tag{1.3}$$

Here, Y_n is the so-called stage vector of dimension sd, whose components Y_{ni} represent approximations to the solution at $t = t_{n-1} + c_i h_n$, where $c := (c_1, \dots, c_s)^T$ is the vector of abscissae and s is the number of Runge–Kutta stages. The vector $F(Y_n)$ contains the derivative values $f(Y_{ni})$. The arrays α, χ, A and G contain method parameters and are of dimension $s \times 1$, $k \times 1$, $s \times s$ and $s \times k$, respectively. These parameters and the abscissae c_i depend on $h^{(n)}$. We remark that a way of circumventing this dependence on $h^{(n)}$ is interpolating the previous steppoints, so that they are equally spaced. However, this strategy adds local errors and does not allow good stepsize flexibility, see [Sch94, p.68].

Stability has been investigated for fixed stepsizes in the literature. Even for large values of k, these methods have "surprisingly" good stability properties [HW91, p.296]. For example, MRKs of Radau type with $s = 3$ remain stiffly stable for $k \leq 28$ and have modest error constants [Sch94, p.13].

A drawback of using MRKs is the high cost of solving the non-linear system (1.3) of dimension sd every time step. Normally, one uses a (modified) Newton process to solve this non-linear system. This leads to a sequence of iterates $Y_n^{(0)}, Y_n^{(1)}, \dots, Y_n^{(m)}$ which are obtained as solutions of the sd-dimensional linear systems

$$(I - A \otimes h_n J_n)(Y_n^{(j)} - Y_n^{(j-1)}) = -R(Y_n^{(j-1)}), \quad j = 1, 2, \dots, m, \tag{1.4}$$

where J_n is the Jacobian of the function f in (1.1) evaluated in t_n, the starting vector $Y_n^{(0)}$ is defined by some predictor formula, and $Y_n^{(m)}$ is accepted as approximation to Y_n. If we use Gaussian elimination to solve these linear systems, then this would cost $\frac{2}{3}s^3d^3$ arithmetic operations for the LU-decompositions.

In order to reduce these costs, one can bring the Newton matrix $I - A \otimes h_n J_n$ to block diagonal form by means of similarity transformations [But76] resulting in

$$\begin{aligned}(I - T^{-1}AT \otimes h_n J_n)(X_n^{(j)} - X_n^{(j-1)}) &= -(T^{-1} \otimes I)R(Y_n^{(j-1)}), \\ Y_n^{(j)} &= (T \otimes I)X_n^{(j)}, \qquad j = 1, 2, \ldots, m.\end{aligned} \tag{1.5}$$

Here, $T^{-1}AT$ is of (real) block diagonal form. Every block of $T^{-1}AT$ corresponds with an eigenvalue pair of A. If the eigenvalue of A is complex, then the block size of the associated block in $T^{-1}AT$ is 2, if the eigenvalue is real, then the block size is 1. The LU-costs are now reduced to $\frac{2}{3}d^3$ and $\frac{16}{3}d^3$ for the blocks of size 1 and 2, respectively. Hairer & Wanner used this approach in their code RADAU5 [HW95]. The blocks of the linear system (1.5) are now decoupled, so that the use of σ processors reduces the effective costs to $\frac{16}{3}d^3$, where σ is the number of blocks in $T^{-1}AT$. Notice that pairs of stage values can be computed concurrently, i.e. it is possible to do function evaluations, transformations and vector updates for pairs of stages in parallel if σ processors are available.

By exploiting the special structure of the $2d$-dimensional linear systems in (1.5), it is possible to reduce the costs of solving these systems (see e.g. [Bin85]). Let $\xi_j \pm i\eta_j$ be an eigenvalue pair and assume that the matrix of the corresponding linear system is of the form

$$\begin{pmatrix} I - \xi_j h_n J_n & -\eta_j h_n J_n \\ \eta_j h_n J_n & I - \xi_j h_n J_n \end{pmatrix}. \tag{1.6}$$

One easily checks that the inverse of (1.6) is

$$(I \otimes \Gamma^{-1}) \begin{pmatrix} I - \xi_j h_n J_n & \eta_j h_n J_n \\ -\eta_j h_n J_n & I - \xi_j h_n J_n \end{pmatrix}, \qquad \Gamma = I - 2\xi_j h_n J_n + (\xi_j^2 + \eta_j^2) h_n^2 J_n^2. \tag{1.7}$$

Using σ processors, the $\mathcal{O}(d^3)$ costs of this approach are $\frac{8}{3}d^3$ ($2d^3$ for the computation of J_n^2 and $\frac{2}{3}d^3$ for the LU-decomposition of Γ). On σ processors, an MRK using this implementation strategy is 4 times more expensive in terms of $\mathcal{O}(d^3)$ costs than a BDF, for which we only have to solve linear systems with a matrix of the form $I - h_n\beta J_n$.

In this paper we reduce the implementational costs of MRKs to a further extent by following the approach of [HS96]. Here, the matrix A is approximated by a matrix B with positive distinct eigenvalues and the iterates $Y_n^{(j)}$ in (1.4) are computed by means of the inner iteration process

$$\begin{aligned}(I - B \otimes h_n J_n)(Y_n^{(j,\nu)} - Y_n^{(j,\nu-1)}) &= -(I - A \otimes h_n J_n)Y_n^{(j,\nu-1)} + C_n^{(j-1)}, \\ C_n^{(j-1)} &:= (I - A \otimes h_n J_n)Y_n^{(j-1)} - R(Y_n^{(j-1)}).\end{aligned} \tag{1.8}$$

The index ν runs from 1 to r and $Y_n^{(j,r)}$ is accepted as the solution $Y_n^{(j)}$ of the Newton process (1.4). Furthermore, $Y_n^{(j,0)} = Y_n^{(j-1)}$. Since the matrix B in (1.8) has distinct eigenvalues, applying a similarity transformation Q that diagonalizes B, i.e. $BQ = QD$ where D is a diagonal matrix, leads to:

$$\begin{aligned}(I - D \otimes h_n J_n)(X_n^{(j,\nu)} - X_n^{(j,\nu-1)}) &= -(I - Q^{-1}AQ \otimes h_n J_n)X_n^{(j,\nu-1)} \\ &\quad +(Q^{-1} \otimes I)C_n^{(j-1)}, \quad \nu = 1, \ldots, r.\end{aligned} \tag{1.9}$$

The system (1.9) consists of s decoupled systems of dimension d which can be solved in parallel. Every processor computes a stage value. The costs for the LU-decompositions are now reduced to $\frac{2}{3}d^3$ on s processors. Notice that in order to ensure the non-singularity of the matrix $(I - D \otimes h_n J_n)$ the positiveness of the eigenvalues of B is required. In analogy with [HS96] we will refer to (1.8) as PILSMRK, Parallel Linear System solver for Multistep Runge–Kutta methods. The combination of modified Newton and PILSMRK will be called the Newton-PILSMRK method.

We will discuss several strategies to choose B such that the inner iterates in (1.8) converge quickly to the Newton iterates in (1.4). Experiments show that, if we apply more than 2 Newton iterations, then only 1 inner iteration suffices to find the Newton iterate. This means that in terms of LU-decompositions and Jacobian evaluations a k-step, s-stage Newton-PILSMRK on s processors is as expensive as a k-step BDF on 1 processor, whereas the stability properties of Newton-PILSMRK are better. If both methods perform the same number of function evaluations, then the accuracies delivered by Newton-PILSMRK are also higher than that of BDF. It turns out that the convergence behaviour of the inner iteration process becomes better if k increases. In particular, the inner iteration process for MRKs converges faster than that for the one-step RK methods proposed in [HS96].

The outline of the paper is as follows. § 2 briefly describes how to determine the MRK parameters. In § 3 we investigate the convergence of the inner iteration process for several choices of the matrix B, and we consider the stability of the overall method in § 4. Numerical experiments in § 5 show the performance of the proposed methods on a number of test problems. Finally, we draw some conclusions in § 6.

2. Construction of MRKs

A large class of multistep Runge-Kutta methods consists of multistep collocation methods, which were first investigated by Guillou and Soulé [GS69]. Later, Lie and Nørsett [LN89] considered the MRKs of Gauss type and Hairer and Wanner [HW91] those of Radau type. In the useful thesis of Schneider [Sch94] on MRKs for stiff ODEs and DAEs a lot of properties of MRKs and further references can be found.

For convenience of the reader we briefly describe here how one can compute c, G and A. Alternative ways of deriving these parameters can be found in [HW91] and [Sch94]. In a multistep collocation method, the solution is approximated by a so-called collocation polynomial. Given $y^{(n)}$, $h^{(n)}$ and c, we define the collocation polynomial $u(t)$ of degree $s+k-1$ by

$$\begin{aligned} u(t_j) &= y_j, && j = n-k+1, \ldots, n, \\ u'(t_n + c_i h_n) &= f(u(t_n + c_i h_n)), && i = 1, \ldots, s. \end{aligned}$$

The stage vector Y_n is then given by $(u(t_n + c_1 h_n)^T, \ldots, u(t_n + c_s h_n)^T)^T$. In order to compute $u(t)$, we expand it in terms of polynomials ϕ_i and ψ_i of degree $s+k-1$, given by

$$\begin{aligned}
\phi_i(\tau_j) &= \delta_{ij}, & j &= 1,\dots,k, & i &= 1,\dots,k,\\
\phi_i'(c_j) &= 0, & j &= 1,\dots,s, & i &= 1,\dots,k,\\
\psi_i(\tau_j) &= 0, & j &= 1,\dots,k, & i &= 1,\dots,s,\\
\psi_i'(c_j) &= \delta_{ij}, & j &= 1,\dots,s, & i &= 1,\dots,s.
\end{aligned}$$

Here, δ_{ij} denotes the Kronecker tensor, τ is the dimensionless coordinate $\frac{t-t_n}{h_n}$ and $\tau_j = \frac{t_{n-k+j}-t_n}{h_n}$, $j = 1,\dots,k$. In terms of these polynomials the expansion of $u(t)$ is given by

$$\begin{aligned}
u(t_n+\tau h) &= \sum_{j=1}^{k} \phi_j(\tau) y_{n-k+j} + h_n \sum_{j=1}^{s} \psi_j(\tau) u'(t_n + c_j h_n)\\
&= \sum_{j=1}^{k} \phi_j(\tau) y_{n-k+j} + h_n \sum_{j=1}^{s} \psi_j(\tau) f(u(t_n + c_j h_n)), \qquad j = 1,\dots,s.
\end{aligned}$$

Clearly, the MRK parameters read $G_{ij} = \phi_j(c_i)$, $A_{ij} = \psi_j(c_i)$, $\alpha_j = \phi_j(1)$ and $\chi = \psi_j(1)$. Notice that the order of the approximations $u(t_n + c_i h_n)$, the so-called *stage order* of the MRK, is $s + k - 1$.

To construct the polynomials $\phi_i(\tau)$ and $\psi_i(\tau)$, we expand them as

$$\phi_i(\tau) = \sum_{m=0}^{s+k-1} d^{\phi}_{m,i} \tau^m \qquad \text{and} \qquad \psi_i(\tau) = \sum_{m=0}^{s+k-1} d^{\psi}_{m,i} \tau^m.$$

Substituting the first expression into the defining conditions yields

$$\begin{pmatrix}
1 & \tau_1 & \tau_1^2 & \tau_1^3 & \dots & \tau_1^{s+k-1}\\
\vdots & \vdots & & \vdots & & \vdots\\
1 & \tau_k & \tau_k^2 & \tau_k^3 & \dots & \tau_k^{s+k-1}\\
0 & 1 & 2c_1 & 3c_1^2 & \dots & (s+k-1)c_1^{s+k-2}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & 1 & 2c_s & 3c_s^2 & \dots & (s+k-1)c_s^{s+k-2}
\end{pmatrix}
\begin{pmatrix} d^{\phi}_{0,i}\\ \vdots \\ d^{\phi}_{s+k-1,i}\end{pmatrix} = e_i. \tag{2.1}$$

The matrix of order $s + k$ in (2.1) will be denoted by W. For the polynomials $\psi_i(\tau)$ we derive analogously

$$W \begin{pmatrix} d^{\psi}_{0,i}\\ \vdots \\ d^{\psi}_{s+k-1,i}\end{pmatrix} = e_{k+i}.$$

To compute the A and the G, we evaluate $\phi_i(\tau)$ and $\psi_i(\tau)$ in $\tau = c_j$ for $j = 1,\dots,s$, yielding

$$\begin{aligned}
\phi_i(c_j) &= (1\ c_j\ \dots c_j^{s+k-1})\ W^{-1} e_i,\\
\psi_i(c_j) &= (1\ c_j\ \dots c_j^{s+k-1})\ W^{-1} e_{i+k}.
\end{aligned}$$

Introducing

$$V = \begin{pmatrix} 1 & c_1 & \dots & c_1^{s+k-1} \\ \vdots & \vdots & & \vdots \\ 1 & c_s & \dots & c_s^{s+k-1} \end{pmatrix},$$

the matrices G and A are respectively given by

$$G = VW^{-1}(e_1, \dots, e_k) \qquad \text{and} \qquad A = VW^{-1}(e_{k+1}, \dots, e_{k+s}).$$

We now construct the abscissae vector c such that we have superconvergence in the steppoints. Only stiffly accurate Multistep Runge–Kutta methods will be considered, i.e. $c_s = 1$. This means that we can omit steppoint formula (1.2) and obtain y_{n+1} from $y_{n+1} = (e_s^T \otimes I)Y_n$. A well known subclass of stiffly accurate MRK methods are the multistep Radau methods, which are $A(\alpha)$-stable. Their set of collocation points $c_1, \dots, c_{s-1}$ is given (see [HW91, p.294]) as the roots in the interval [0,1] of

$$\sum_{j=1}^{k} \frac{1}{c_i - \tau_j} + \sum_{\substack{j=1 \\ j \neq i}}^{s} \frac{2}{c_i - c_j} = 0, \quad i = 1, \dots, s-1.$$

We call the order of approximation y_{n+1} to $y(t_{n+1})$ the *steppoint order* or, more loosely, the *order* of the MRK. This choice of c leads to steppoint order $2s + k - 2$.

The appendix to this paper lists the MRK parameters for $s \in \{2, 4\}$ and $k \in \{2, 3\}$.

3. Convergence of the inner iteration process

We now discuss the choice of the matrix B in (1.8) such that the inner iteration process converges rapidly. If we define the *inner iteration error* by $\epsilon_n^{(j,\nu)} := Y_n^{(j,\nu)} - Y_n^{(j)}$, then (1.4) and (1.8) yield the recursion

$$\epsilon_n^{(j,\nu)} = Z(h_n J_n)\epsilon_n^{(j,\nu-1)}, \qquad Z(h_n J_n) := (I - B \otimes h_n J_n)^{-1}((A - B) \otimes h_n J_n).$$

Applying the method to Dahlquist's test equation

$$y' = \lambda y, \quad \lambda \in \mathbf{C}, \tag{3.1}$$

this recursion reduces to

$$\epsilon_n^{(j,\nu)} = Z(z_n)\epsilon_n^{(j,\nu-1)}, \qquad z_n := h_n \lambda. \tag{3.2}$$

Let $\mu(\cdot)$ be the logarithmic norm associated with the Euclidean norm, which can be expressed as $\mu(S) := \frac{1}{2}\lambda_{\max}(S + S^T)$, where $\lambda_{\max}(\cdot)$ denotes the algebraically largest eigenvalue of a matrix (see e.g. [HNW93, p.61]). For dissipative problems $\mu(J_n) \leq 0$. The following lemma states that the inner iteration process converges for dissipative problems at least as fast as for the 'most unfavourable' linear test equation. For the proof of this lemma we refer to [Nev85].

Lemma 1 *If* $\mu(J_n) \leq 0$, *then* $\|Z(h_n J_n)^\nu\|_2 \leq \max\{\|Z^\nu(z_n)\|_2 : Re(z_n) \leq 0\}$.

In § 3.1 and § 3.2 we treat two choices for the matrix B that make $Z(z_n)$ 'small' in some sense. To measure $Z(z_n)$ we use the following quantities:

- $\rho^{(j)}(z_n)$, the (averaged) rate of convergence after j iterations in z_n, defined by

$$\rho^{(j)}(z_n) := \sqrt[j]{\|Z(z_n)^j\|_2}.$$

- $\rho^{(j)}_\infty$, the stiff convergence rate after j iterations, defined by

$$\rho^{(j)}_\infty := \sqrt[j]{\|Z^j_\infty\|_2}, \quad Z_\infty := \lim_{z_n \to \infty} Z(z_n) = (I - B^{-1}A).$$

  Z_∞ will be referred to as the *stiff amplification matrix.*

- $\rho^{(j)}$, the maximal convergence rate after j iterations, defined by

$$\rho^{(j)} := \max_{\mathrm{Re}(z_n)\leq 0} \{\rho^{(j)}(z_n)\}.$$

Notice that because of the maximum principle and the fact that $\rho^{(j)}(z_n)$ is symmetric with respect to the real axis, $\rho^{(j)}(z_n)$ takes its maximum at the positive imaginary axis:

$$\rho^{(j)} := \max_{(x_n)\geq 0} \{\rho^{(j)}(\mathrm{i}x_n)\}.$$

Since A depends on $h^{(n)}$, B also depends on $h^{(n)}$. Consequently, the procedure for constructing B has to be carried out every time $h^{(n)}$ changes and should not be too expensive.

3.1 Constructing B: Crout decomposition

Let L be the lower triangular matrix of the Crout decomposition of A, i.e. L is lower triangular such that $L^{-1}A$ is upper triangular with ones on the diagonal. As proposed in [HS95], we choose $B = L$. The stiff amplification matrix takes the form $I - L^{-1}A$, which is strictly upper triangular. Consequently, $\rho^{(j)}_\infty = 0$ for $j \geq s$. For reasons that will become clear in § 3.2, we will refer to this inner iteration process as PILSMRK(L,I).

Table 1 lists the values of $\rho^{(j)}$ for a few PILSMRK(L,I) methods for the case with constant stepsizes. As a reference we included the one-step Radau IIA methods. From this table we see that, for the worst-case situation, the convergence of the MRKs is better than that of the one-step Runge–Kutta methods.

In practice, the rate of convergence in other points of the complex plane is also of interest. Figure 1 shows $\rho^{(j)}(z_n)$ along the imaginary axis $z_n = \mathrm{i}x_n$, $x_n \in \mathbf{R}$ for PILSMRK(L,I) with ($k = 3, s = 4$) method with constant stepsizes for $j = 1, 2, 3, 4$ and $j = \infty$. From this figure we clearly see that $\rho^{(j)}_\infty = 0$ for $j \geq s$.

In order to see the effect of variable stepsizes on the convergence rate, we define

Table 1: Values of $\rho^{(j)}$ for several PILSMRK(L,I) methods with constant stepsizes.

s	k	Order	$j=1$	$j=2$	$j=3$	$j=4$	...	$j=\infty$
2	1	3	0.24	0.21	0.20	0.19	...	0.18
	2	4	0.19	0.17	0.16	0.16	...	0.15
	3	5	0.17	0.15	0.15	0.14	...	0.14
4	1	7	0.59	0.54	0.53	0.52	...	0.51
	2	8	0.54	0.50	0.49	0.48	...	0.47
	3	9	0.52	0.48	0.47	0.46	...	0.44
8	1	15	1.03	0.94	0.91	0.90	...	0.86
	2	16	0.98	0.92	0.89	0.88	...	0.84
	3	17	0.97	0.92	0.89	0.87	...	0.82

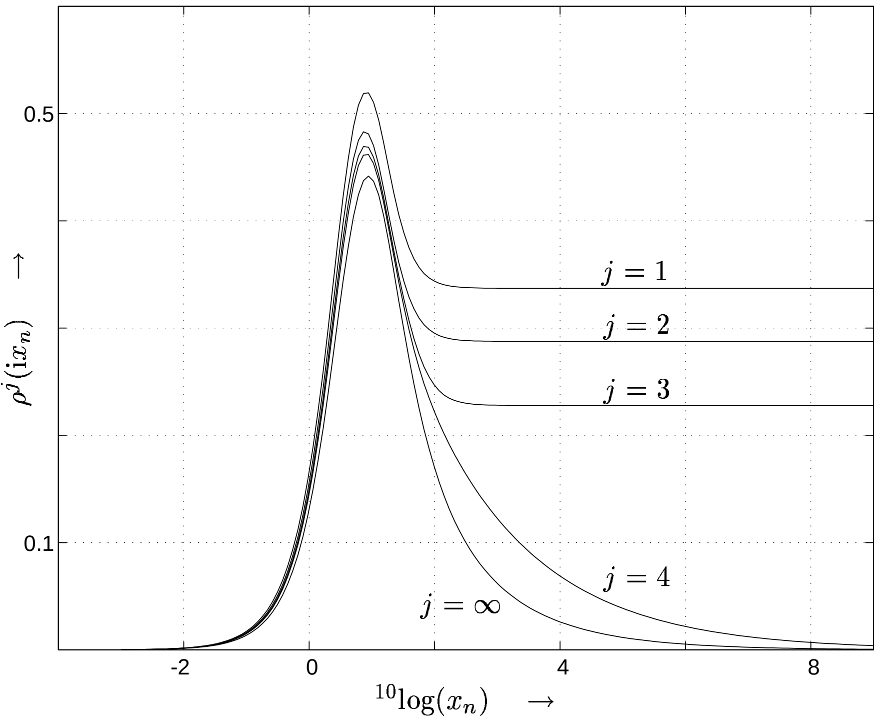

Figure 1: $\rho^{(j)}(\mathrm{i}x_n)$ for PILSMRK(L,I) with $k=3, s=4$.

$$\omega_i = h_i/h_{i-1} \quad \text{for} \quad i = n-k+2,\dots,n$$

and plotted $\rho^{(j)}$ as function of ω_i for several PILSMRK methods. Here, $\omega_i \in [0.2, 2]$, since in an actual implementation, a reasonable factor by which subsequent stepsizes are multiplicated lies in this interval. These plots revealed that the influence of variable stepsizes on the rate of convergence is modest. E.g., for $k=2$, $s=4$, $\rho^{(j)} \in [0.45, 0.58]$, $\forall j$, and for $k=3$, $s=4$, $\rho^{(j)} \in [0.495, 0.525]$, $\forall j$.

3.2 Constructing B: Schur-Crout decomposition

Before approximating the matrix A by the lower factor of the Crout decomposition, we first transform A to 'a more triangular form', the real Schur form. In the usual eigenvalue problem the eigenvalues and eigenvectors are unknown. However, the problem with which we are faced here is computing a real Schur form, given the eigenvalues and eigenvectors of A. Below we specify precisely how we do this. We remark that this construction is not developed to be cheap, but such that we are able to exploit the freedom in the real Schur form.

Let γ be the vector with eigenvalues of A, and ξ and η be the real and imaginary part of γ, respectively, i.e. $\gamma = \xi + \mathrm{i}\eta$. Order the components of γ as follows (we will motivate this choice later):

$$|\eta_i^2/\xi_i| \geq |\eta_j^2/\xi_j| \quad \text{for} \quad i > j. \tag{3.3}$$

In addition, if $|\eta_i^2/\xi_i| = |\eta_{i+1}^2/\xi_{i+1}|$, then $\eta_i > 0$. This sequence is such that real eigenvalues have the lowest index in γ and complex eigenvalues are ordered in conjugated pairs by increasing value of η_j^2/ξ_j, while the eigenvalue with positive imaginary part comes first within a pair. The matrices A treated in this paper have at most one real eigenvalue, so that we do not have to sort real eigenvalues.

Let $e_j^{\mathrm{r}} + \mathrm{i}e_j^{\mathrm{i}}$ be the eigenvector belonging to γ_j, such that $\|e_j^{\mathrm{r}} + \mathrm{i}e_j^{\mathrm{i}}\|_2 = 1$ and $e_{j1}^{\mathrm{i}} = 0$. For all matrices A that are of interest here, this scaling turns out to exist. Define

$$E = (\; e_1^{\mathrm{r}} \quad e_1^{\mathrm{i}} \quad e_3^{\mathrm{r}} \quad e_3^{\mathrm{i}} \quad \ldots \quad e_{s-1}^{\mathrm{r}} \quad e_{s-1}^{\mathrm{i}} \;)$$

if A has only complex eigenvalues, and

$$E = (\; e^{\mathrm{r}} \quad e_2^{\mathrm{r}} \quad e_2^{\mathrm{i}} \quad e_4^{\mathrm{r}} \quad e_4^{\mathrm{i}} \quad \ldots \quad e_{s-1}^{\mathrm{r}} \quad e_{s-1}^{\mathrm{i}} \;)$$

if A has one real eigenvalue with eigenvector e^{r}. One easily verifies that the matrix $E^{-1}AE$ is block diagonal with 2×2 blocks

$$\begin{pmatrix} \xi_j & \eta_j \\ -\eta_j & \xi_j \end{pmatrix},$$

and one block equal to ξ_1 if $\eta_1 = 0$. We orthonormalize the columns of E by a Gram-Schmidt process, i.e. we construct a lower block triangular matrix K such that EK is orthogonal. This matrix EK transforms A to a matrix H:

$$H := (EK)^{-1}A(EK) = K^{-1}(E^{-1}AE)K. \tag{3.4}$$

Since K is lower triangular and $E^{-1}AE$ is block diagonal, it is clear that H is lower block triangular. Notice that H is a real Schur form of A.

We now rotate the diagonal blocks of H by means of a matrix Θ such that $\Theta^{-1}H\Theta$ is 'more suitable' to be approximated by its lower Crout factor. Define

$$\Theta = \mathrm{diag}(\Theta_j), \qquad \Theta_j = \begin{pmatrix} \cos\theta_j & \sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{pmatrix}, \qquad \Theta_1 = 1 \quad \text{if} \quad \eta_1 = 0,$$

and $S = \Theta^{-1}H\Theta$. Here, $j \in \{2, 4, \ldots, s-1\}$ if $\eta_1 = 0$ and $j \in \{1, 3, \ldots, s-1\}$ if $\eta_1 \neq 0$. The lower factor of the Crout decomposition of S is again denoted by L. Remark that the stiff amplification matrix $I - L^{-1}S$ is block diagonal with 2×2 blocks containing only one non-zero entry. One easily verifies that this entry is given by

$$-S_{j,j+1}/S_{j,j}, \tag{3.5}$$

where

$$\begin{aligned} S_{j,j} &= \frac{1}{2}((H_{j,j}-H_{j+1,j+1})\cos(2\theta_j) - (H_{j,j+1}+H_{j+1,j})\sin(2\theta_j) + (H_{j,j}+H_{j+1,j+1})), \\ S_{j,j+1} &= \frac{1}{2}((H_{j+1,j}+H_{j,j+1})\cos(2\theta_j) + (H_{j,j}-H_{j+1,j+1})\sin(2\theta_j) + (H_{j,j+1}+H_{j+1,j})), \end{aligned}$$

and the diagonal blocks of H and S are of the form

$$\begin{pmatrix} H_{j,j} & H_{j,j+1} \\ H_{j+1,j} & H_{j+1,j+1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} S_{j,j} & S_{j,j+1} \\ S_{j+1,j} & S_{j+1,j+1} \end{pmatrix}.$$

We choose θ_j such that the absolute value of (3.5) is minimized. By using Maple [CGG$^+$91] we established that this is done for

$$\theta_j = \arctan \frac{H_{j,j}H_{j+1,j} + H_{j,j+1}H_{j+1,j+1} + \sqrt{\det(H)(\|H\|_F^2 - 2\det(H))}}{H_{j+1,j}^2 + H_{j+1,j+1}^2 - \det(H)} \bmod \pi,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The matrix B with real eigenvalues that approximates A is thus given by $B = ULU^T$, where $U := EK\Theta$ is an orthogonal matrix. Applying a similarity transformation Q such that $BQ = QD$, we again arrive at scheme (1.9). The linear system solver resulting from this Schur-Crout approach will be referred to as PILSMRK(L,U), where the U indicates that we have transformed A before approximating it by L.

We now illustrate the idea that moved us to sort the eigenvalues as in (3.3). For simplicity of notation, we assume here that $s = 4$. If the first order expansion of $Z(z_n)$ for small z_n is given by

$$Z(z_n) := z_n Z_0 + \mathcal{O}(z_n^2),$$

then $Z_0 = A - B$. It can be verified that for the Schur-Crout approach Z_0 is of the form

$$Z_0 = U \begin{pmatrix} 0 & \zeta_{12} & 0 & 0 \\ 0 & \zeta_{22} & 0 & 0 \\ 0 & \zeta_{32} & 0 & \zeta_{34} \\ 0 & \zeta_{42} & 0 & \zeta_{44} \end{pmatrix} U^T,$$

Table 2: Values of $\rho^{(j)}$ for several PILSMRK(L,U) methods with constant stepsizes.

s	k	Order	$j=1$	$j=2$	$j=3$	$j=4$	...	$j=\infty$
2	1	3	0.24	0.21	0.20	0.19	...	0.18
	2	4	0.18	0.16	0.16	0.15	...	0.15
	3	5	0.15	0.14	0.13	0.13	...	0.13
4	1	7	0.55	0.49	0.47	0.47	...	0.44
	2	8	0.50	0.45	0.43	0.43	...	0.41
	3	9	0.47	0.42	0.41	0.40	...	0.39
8	1	15	0.91	0.78	0.74	0.72	...	0.65
	2	16	0.88	0.76	0.72	0.70	...	0.62
	3	17	0.86	0.74	0.70	0.68	...	0.61

where

$$\begin{pmatrix} \zeta_{32} \\ \zeta_{42} \end{pmatrix} = v \begin{pmatrix} S_{31} \\ S_{41} \end{pmatrix}, \quad v = -\frac{1}{S_{21}} \frac{\eta_1^2}{\xi_1}.$$

In order to keep the lower triangular part of Z_0 as small as possible, the best we can do is sorting the eigenvalues such that those with the smallest value of η_k^2/ξ_k come first.

Table 2 and Figure 2 are the analogues of Table 1 and Figure 1 for PILSMRK(L,U). The worst-case $\rho^{(j)}$-values in Table 2 are smaller than those in Table 1. The difference between PILSMRK(L,I) and PILSMRK(L,U) becomes larger in favour of PILSMRK(L,U) as s increases. This can be understood by realizing that for the Crout option, we approximate the matrix A with s^2 parameters by a matrix B with $s(s+1)/2$ entries, whereas for the Schur-Crout case, the matrix U^TAU with $s(s+1)/2 + l$ nonzero entries, where l is the number of complex conjugated eigenvalue pairs, is approximated by U^TBU with $s(s+1)/2$ parameters. In addition, the advantage of PILSMRK(L,U) over PILSMRK(L,I) is that the stiff convergence rate $\rho_\infty^{(j)}$ vanishes for $j > 1$, which is confirmed by Figure 2. The extra price that we have to pay is the construction of the real Schur decomposition of A every time ω_j changes for some j. Since in practice $s \ll d$, we do not consider this as a serious drawback.

Remark 1 There is freedom in the choice of the transformation matrix Q that diagonalizes B. If X is a matrix with eigenvectors of B and Σ and P are diagonal and permutation matrices, respectively, then for every matrix Q of the form

$$Q = X\Sigma P, \tag{3.6}$$

we have that $BQ = QD$. Starting with a fixed matrix X, we determined Σ and P in (3.6) such that the elements of Q and Q^{-1} are not too large. □

Remark 2 Another approach for finding a suitable matrix B, based on rotations that minimize $\rho^{(1)}$, can be found in [HM96]. □

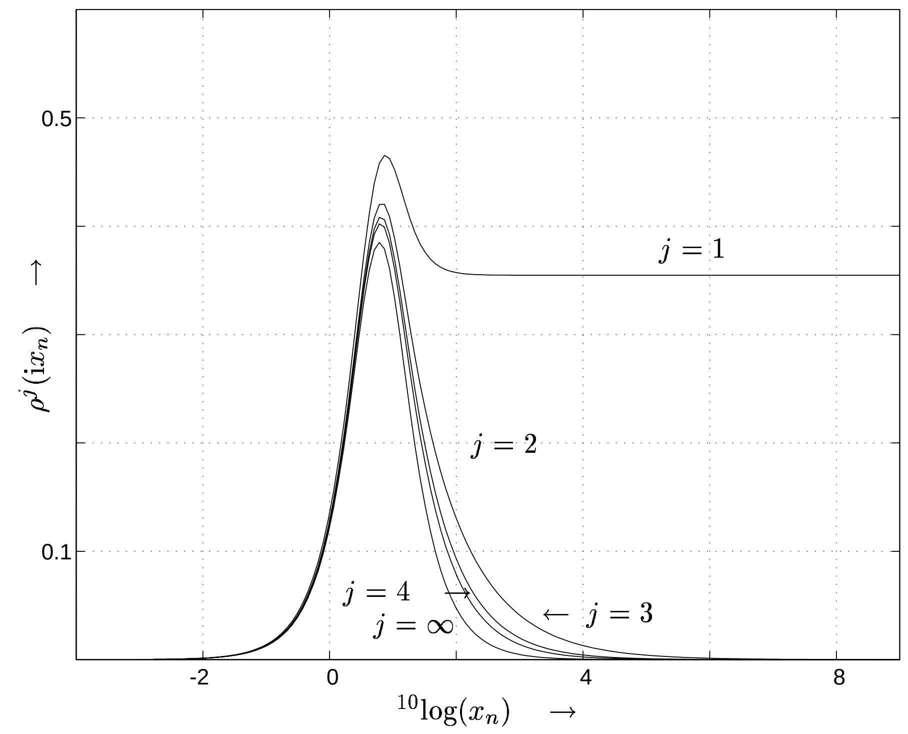

Figure 2: $\rho^{(j)}(\mathrm{i}x_n)$ for PILSMRK(L,U) with $k = 3, s = 4$.

The matrices D and Q that result from the Crout and Schur-Crout approaches are given in the appendix to this paper for several values of k and s.

4. Stability

In this paragraph we investigate the stability of the corrector formula (1.3) and the PILSMRK method (1.8) for test equation (3.1) solved with constant stepsizes h. We only consider stiffly accurate methods, i.e. $y_n = e_s^T Y_n^{(m,r)}$.

Following [Sch94] we write (1.3) in the form

$$y^{(n)} = M(z)y^{(n-1)}, \qquad M(z) = \begin{pmatrix} N \\ e_s^T(I - zA)^{-1}G \end{pmatrix}, \qquad z := h\lambda,$$

where the $(k-1) \times k$ matrix N is given by

$$N = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}.$$

The *stability region* is defined by

$$\mathcal{S} := \{z \in \mathbf{C} \mid \rho(M(z)) < 1\}, \tag{4.1}$$

where $\rho(\cdot)$ denotes the spectral radius function. We use the quantity $\widehat{D}^{(mr)}$ to measure the stability region (see [HW91, p.268]), where

$$\widehat{D} := -\inf\{\mathrm{Re}(z) \mid z \notin \mathcal{S}\}.$$

In practice, the PILSMRK method will be used to solve the corrector only approximately. Therefore we do not attain the stability of the corrector. For conducting a stability analysis for the PILSMRK methods we assume that in each step m outer and r inner iterations are carried out. In addition we assume that the predictor is only based on the stage vector in the previous steppoint,

$$Y_n^{(0,r)} = (P \otimes I) Y_{n-1}^{(m,r)}, \tag{4.2}$$

where P is an $s \times s$ matrix. From (3.2) and (1.3) we derive a recursion in ν:

$$Y_n^{(j,\nu)} = Z(z) Y_n^{(j,\nu-1)} + (I - zB)^{-1} G y^{(n-1)}.$$

An elementary manipulation, in which we use $Y_n^{(j,0)} = Y_n^{(j-1,r)}$, leads to a recursion in j:

$$Y_n^{(j,r)} = Z^r(z) Y_n^{(j-1,r)} + (I - Z^r(z))(I - zA)^{-1} G y^{(n-1)}.$$

Substituting (4.2) yields the following recursion in time:

$$Y_n^{(m,r)} = Z^{mr}(z) P Y_{n-1}^{(m,r)} + (I - Z^{mr}(z))(I - zA)^{-1} G y^{(n-1)}, \tag{4.3}$$

which we write in the form

$$\begin{pmatrix} y^{(n)} \\ Y_n^{(m,r)} \end{pmatrix} = M^{(mr)}(z) \begin{pmatrix} y^{(n-1)} \\ Y_{n-1}^{(m,r)} \end{pmatrix}, \qquad M^{(mr)}(z) = \begin{pmatrix} M_{11}^{(mr)}(z) & M_{12}^{(mr)}(z) \\ M_{21}^{(mr)}(z) & M_{22}^{(mr)}(z) \end{pmatrix}.$$

From (4.3) we see that

$$M_{21}^{(mr)}(z) = (I - Z^{mr}(z))(I - zA)^{-1} G, \qquad M_{22}^{(mr)}(z) = Z^{mr}(z) P.$$

Since we restrict ourselves here to stiffly accurate methods,

$$M_{11}^{(mr)} = \begin{pmatrix} N \\ e_s^T M_{21}^{(mr)} \end{pmatrix}, \qquad M_{12}^{(mr)} = \begin{pmatrix} O_{k-1,s} \\ e_s^T M_{22}^{(mr)} \end{pmatrix},$$

where O_{ij} denotes an $i \times j$ zero matrix. Notice that this linear stability analysis does not distinguish between outer and inner iterations. In analogy with (4.1) we define the *stability region after mr iterations* by

Table 3: Values of $\widehat{D}^{(mr)}$ for PILSMRK(L,I) with k steps and s stages.

s	k	$mr=1$	$mr=2$	$mr=4$	$mr=6$	$mr=8$	$mr=10$	$mr=20$	$mr=\infty$
2	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0.0094	0.0823	0.0838	0.0838	0.0838	0.0838	0.0838
	4	0	0.3435	0.4601	0.4610	0.4610	0.4610	0.4610	0.4610
4	1	$*$	$*$	0	0	0	0	0	0
	2	$*$	$*$	0	0	0	0	0	0
	3	$*$	$*$	0	0	0.0006	0.0021	0.0025	0.0025
	4	$*$	$*$	0	0	0.0120	0.0180	0.0192	0.0192
8	1	0	0	0.0677	0.0480	0.0239	0.0103	0	0
	2	0	0	0.0624	0.0405	0.0188	0.0076	0	0
	3	0	0	0.0590	0.0363	0.0162	0.0064	0	0
	4	0	0	0.0565	0.0335	0.0145	0.0057	0.0004	0.0003

$$\mathcal{S}^{(mr)} := \{z \in \mathbf{C} \mid \rho(M^{(mr)}(z)) < 1\}$$

and the stability measure

$$\widehat{D}^{(mr)} := -\inf\{\mathrm{Re}(z) \mid z \notin \mathcal{S}^{(mr)}\}.$$

It is clear that

$$\lim_{mr\to\infty} \widehat{D}^{(mr)} = \widehat{D}.$$

Table 3 and 4 list $\widehat{D}^{(mr)}$-values for the k-step s-stage MRK of Radau type for $k \in \{1,2,3,4\}$ and $s \in \{2,4,8\}$ with PILSMRK(L,I) and PILSMRK(L,U), respectively. For $s \leq 4$, we used the predictor that extrapolates the previous stage values, i.e. we determined P in (4.2) such that $Y_n^{(0,r)}$ has maximal order. Since extrapolating 8 stages leads to very large entries in P, the predictor for the 8-stage methods was chosen to be the last step value predictor. If $\widehat{D}^{(mr)} > 4$, then this is indicated by $*$.

The $\widehat{D}^{(mr)}$-values for BDF are independent of mr, because for the linear test problem the corrector equation is solved within 1 iteration. For $k = 1,2,3$ and 4 these values are 0, 0, 0.0833 and 0.6665, respectively.

From these tables we see that for $s \leq 4$ the stability of PILSMRK(L,I) is better than that of PILSMRK(L,U). For $s = 8$ the $\widehat{D}$-values are comparable. Relatively to its order, the stability of PILSMRK is much better than that of BDF. As expected, we see that increasing s and decreasing k improves the stability of MRK. If we solve the corrector equation only approximately, then sometimes the stability of the resulting method is even better than that of MRK. For $s = 4$ and $mr \leq 2$, the method is not stable, due to the extrapolation predictor, which is very unstable as stand-alone method. Notice that the $\widehat{D}^{(\infty)}$-values are the values for the underlying MRK corrector.

Table 4: Values of $\widehat{D}^{(mr)}$ for PILSMRK(L,U) with k steps and s stages.

s	k	$mr=1$	$mr=2$	$mr=4$	$mr=6$	$mr=8$	$mr=10$	$mr=20$	$mr=\infty$
2	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0.0216	0.0827	0.0838	0.0838	0.0838	0.0838	0.0838
	4	0	0.3762	0.4605	0.4610	0.4610	0.4610	0.4610	0.4610
4	1	$*$	$*$	0.2214	0	0	0	0	0
	2	$*$	$*$	0.2239	0	0.0001	0	0	0
	3	$*$	$*$	0.2784	0	0.0031	0.0030	0.0025	0.0025
	4	$*$	$*$	0.3474	0.0001	0.0169	0.0194	0.0192	0.0192
8	1	0	0.1060	0.0636	0.0254	0.0212	0.0101	0	0
	2	0	0.1056	0.0557	0.0227	0.0179	0.0080	0	0
	3	0	0.1051	0.0510	0.0210	0.0161	0.0075	0	0
	4	0	0.1046	0.0477	0.0199	0.0152	0.0075	0.0003	0.0003

To get an idea of the shape of $\mathcal{S}^{(mr)}$, Figure 3 shows $\mathcal{S}^{(mr)}$ for PILSMRK(L,U) with 3 steps and 4 stages, where $mr \in \{3, 5, \infty\}$.

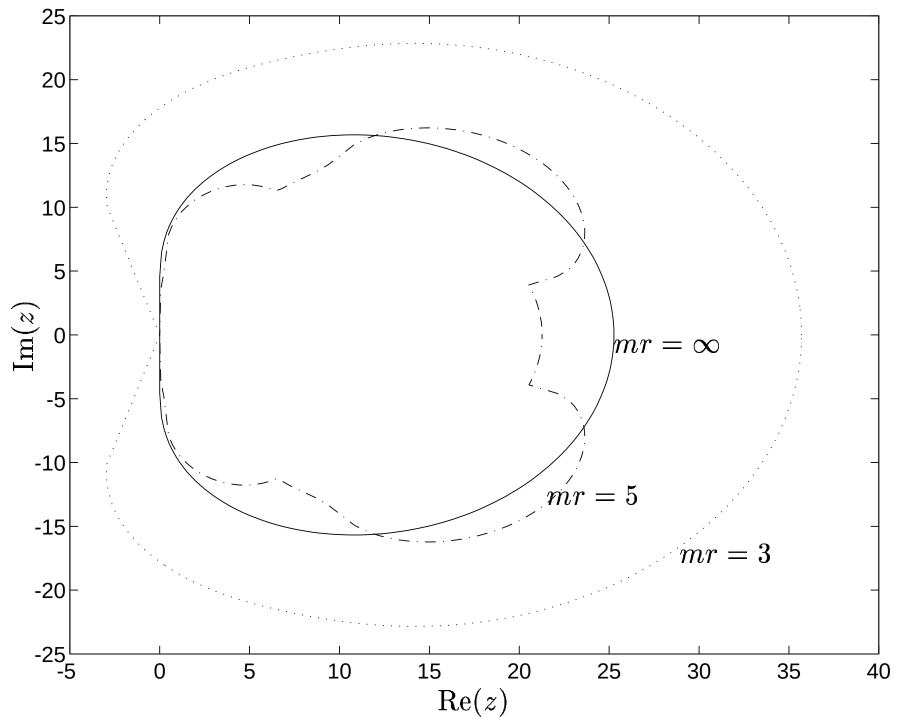

Figure 3: $\mathcal{S}^{(mr)}$ for PILSMRK(L,U) with $k = 3, s = 4$.

Table 5: Results of PILSMRK(L,I) on test problems.

			HIRES					Ring Modulator				
s	k	r	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$
2	2	1	3.3	3.7	4.2	4.7	4.9	$*$	2.8	3.9	3.8	3.8
		2	3.2	3.8	4.3	5.0	4.9	$*$	3.6	3.8	3.8	3.8
		10	3.2	3.8	4.3	5.0	4.9	$*$	3.6	3.8	3.8	3.8
	3	1	3.3	3.7	4.2	4.6	5.2	$*$	3.0	4.1	4.2	4.3
		2	3.2	3.8	4.3	4.8	5.2	$*$	3.8	4.2	4.3	4.3
		10	3.2	3.8	4.3	4.8	5.2	$*$	3.8	4.2	4.3	4.3
4	2	1	$*$	4.6	4.8	5.1	7.3	$*$	$*$	6.1	6.5	8.2
		2	$*$	4.3	4.9	5.3	7.9	$*$	$*$	5.8	6.5	8.2
		10	3.7	4.4	4.9	5.4	7.9	$*$	$*$	5.8	6.4	8.2
	3	1	$*$	4.6	4.8	5.1	7.2	$*$	$*$	6.1	6.5	8.1
		2	$*$	4.3	4.9	5.3	7.8	$*$	$*$	5.8	6.5	8.1
		10	3.7	4.4	4.9	5.4	7.8	$*$	$*$	5.8	6.4	8.1

5. Numerical experiments

In this paragraph we compare several Newton-PILSMRK methods with BDF. We also investigate how many inner iterations PILSMRK needs to find the Newton iterate. Although in practice one would use variable stepsizes and variable order, for these purposes it is sufficient to conduct experiments with fixed stepsize and fixed values of s and k.

Two problems from the 'Test Set for IVP Solvers' [LSV96] are integrated. Our first test example is a problem of Schäfer (called the HIRES problem in [HW91, p.157]) and consists of 8 mildly-stiff non-linear equations on the interval $[5, 305]$. (We adapted the initial condition here such that the integration starts outside the transient phase.) We used stepsize $h = 15$. The second test problem originates form circuit analysis and describes a ring modulator. We integrate this highly stiff system of 15 equations on the interval $[0, 10^{-3}]$ with stepsize $h = 2.5 \cdot 10^{-7}$. Horneber [Hor76] provided this problem.

For $s > 1$ we implemented the extrapolation predictor as defined before, i.e. based on the previous stage vector. For BDF we used the last steppoint value as predictor. We tried extrapolation of more steppoints, but this did not give satisfactory results for both test problems. The starting values $y_1, y_2, \ldots, y_{k-1}$ were obtained using the 8-stage Radau IIA method, in order to be sure that the integration is not influenced by some starting procedure. In the implementation of BDF we solved the non-linear equation of dimension d with modified Newton, using m iterations per time step.

In the tables we list the minimal number of correct digits cd of the components of the numerical solution in the endpoint, i.e. at the endpoint, the absolute errors are written as 10^{-cd}. Negative cd-values are indicated with $*$. The numbers of stages, steps, inner and outer iterations are given by s, k, r and m, respectively.

The tables clearly show that the PILSMRK iterates for $r = 1$ are (almost) of the same quality as the Newton iterates, provided that we perform more than 2 Newton iterations. We

Table 6: Results of PILSMRK(L,U) on test problems.

			HIRES					Ring Modulator				
s	k	r	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$
2	2	1	3.3	3.8	4.2	4.8	4.9	$*$	2.8	3.9	3.8	3.8
		2	3.2	3.8	4.3	5.0	4.9	$*$	3.6	3.8	3.8	3.8
		10	3.2	3.8	4.3	5.0	4.9	$*$	3.6	3.8	3.8	3.8
	3	1	3.3	3.8	4.2	4.7	5.2	$*$	3.1	4.1	4.3	4.3
		2	3.2	3.8	4.3	4.8	5.2	$*$	3.8	4.2	4.3	4.3
		10	3.2	3.8	4.3	4.8	5.2	$*$	3.8	4.2	4.3	4.3
4	2	1	$*$	$*$	4.9	5.1	7.2	$*$	$*$	5.8	6.3	8.2
		2	2.6	4.4	4.9	5.4	7.9	$*$	$*$	5.8	6.4	8.2
		10	3.7	4.4	4.9	5.4	7.9	$*$	$*$	5.8	6.4	8.2
	3	1	$*$	$*$	4.9	5.2	7.2	$*$	$*$	5.8	6.3	8.1
		2	3.6	4.4	4.9	5.4	7.8	$*$	$*$	5.8	6.4	8.1
		10	3.7	4.4	4.9	5.4	7.8	$*$	$*$	5.8	6.4	8.1

Table 7: Results of BDF on test problems.

			HIRES					Ring Modulator				
s	k	r	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$	$m=1$	$m=2$	$m=3$	$m=4$	$m=10$
1	2	1	2.9	3.5	3.1	3.0	3.0	1.1	1.1	1.1	1.1	1.1
	3	1	2.8	3.7	3.6	3.4	3.3	1.6	1.5	1.6	1.6	1.6
	4	1	2.8	3.4	4.4	3.8	3.6	1.8	1.9	1.9	1.9	1.9
	5	1	2.7	3.3	4.2	4.1	3.8	2.4	2.9	2.9	2.9	2.9
	6	1	2.8	3.4	4.1	3.9	3.7	2.4	$*$	2.9	$*$	2.9

also see that Newton-PILSMRK reaches higher accuracies than BDF for the same number of Newton iterations. However, if we want to solve the corrector equation entirely, one would have to perform more Newton iterations for Newton-PILSMRK than for BDF, since the latter is of lower order. Solving the ring modulator, BDF suffers from stability problems for $k = 6$, whereas the methods with $k \leq 4$ give cd-values, that might be too low in practice. For the 4-stage Newton-PILSMRK, the $k = 3$ results are not better than the $k = 2$ results. Performing not more then 10 Newton iterations, which is not sufficient to solve the corrector equation, is responsible for this. Experiments confirmed that using more than 10 iterations for the 3-step 4-stage MRK yields higher accuracies than for the 2-step 4-stage method. A comparison of Table 5 with Table 6 shows that the performance of PILSMRK(L,U) is comparable to that of PILSMRK(L,I). Although PILSMRK(L,U) converges faster than PILSMRK(L,I), the latter has better stability properties for $s \leq 4$. Apparently, these effects neutralize each other for these test problems. However, Tables 1-4 indicate that PILSMRK(L,U) can become better than PILSMRK(L,I) for $s > 4$.

In order to show how the Newton-PILSMRK method performs on an s-processor computer, we implemented the 3-step 4-stage Newton PILSMRK(L,I) on the Cray C-98/4256 at SARA and integrated the ring modulator, using again 4000 constant integration steps. The Cray

Table 8: Speed-up factor of 3-step 4-stage Newton-PILSMRK(L,I) for ring modulator.

	$m = 3$	$m = 4$	$m = 10$
Actual speed-up	3.3	3.3	3.2
Optimal speed-up	3.9	3.9	3.9

C98/4256 is a shared memory computer with four processors. Table 8 lists the speed-up factors of the runs on four processors with respect to the runs in one-processor mode. Since we did not have the machine in dedicated mode during our experiments (on the average we used 2.5 processors concurrently), we used a tool called ATExpert [Cra94b] to predict the actual speed-up factors on four processors. In practice these values turn out to be very reliable. Denoting the fraction of the code that can be done in parallel by f_P, the optimal speed-up on N processors according to Amdahl's law is given by the formula $1/(1 - f_P + f_P/N)$. ATExpert produces these optimal speed-up values, based on estimates of the parallel fraction f_P. These values are also listed in Table 8.

We compiled the codes using the flags `-dp`, `-ZP` and `-Wu"-p"`. The environment variables `NCPUS` and `MP_DEDICATED` were valued 4 and 1, respectively. We refer to the Cray C90 documentation [Cra94a] for the specification of these settings.

From Table 8 we conclude that the Newton-PILSMRK methods have a satisfactory parallel performance.

6. Summary and conclusions

In this paper we proposed the Newton-PILSMRK method, which is a combination of a Newton process applied to a Multistep Runge–Kutta method with a Parallel Iterative Linear System solver. The non-linear equations that arise in an MRK are usually solved by a modified Newton process, in which we have to solve linear systems of dimension sd, where s is the number of Runge–Kutta stages of the MRK and d the dimension of the problem. PILSMRK computes the solutions of these linear systems by means of an inner iteration process, in which we solve s decoupled systems of dimension d. To achieve this decoupling, we have to approximate a matrix A with complex eigenvalues by a matrix B with positive distinct eigenvalues. It turns out that

- the most efficient parallel implementation of an MRK with a Newton process is 4 times more expensive than Newton-PILSMRK on s processors in terms of $\mathcal{O}(d^3)$ costs.
- if we apply more than 2 Newton iterations, then in practice PILSMRK with only 1 inner iteration often suffices to find the Newton iterate,
- in terms of Jacobian evaluations and LU-decompositions, the k-step s-stage Newton-PILSMRK on s processors is equally expensive as the k-step BDF on 1 processor, whereas the order is higher and the stability properties are better than that of BDF,

- for the same number of function evaluations, Newton-PILSMRK delivers higher accuracies than BDF, although Newton-PILSMRK did not solve the corrector equation entirely,
- increasing the number of previous steppoints k, leads to a better convergence behaviour of PILSMRK, but worse stability properties of the MRK,
- in a linear stability analysis, performing more than 3 iterations (inner or outer) suffices to attain at least the stability of the MRK corrector, if $s \leq 4$.
- of the two options proposed here for choosing the matrix B, Crout and Schur-Crout, the latter has a better convergence behaviour, but its stability properties are worse for $s \leq 4$.

Acknowledgements
The authors are grateful to Prof. dr. P.J. van der Houwen for his careful reading of the manuscript and for suggesting several improvements.

Appendix
In this appendix we list the parameters c, G and A in (1.3) for the k-step s-stage MRK method of Radau type for $k \in \{2,3\}$ and $s \in \{2,4\}$. Moreover, we provide the PILSMRK parameters δ and Q, where δ =diag(D) and D, Q are the matrices in (1.9), for both the Crout approach (i.e. PILSMRK(L,I)) and the Schur-Crout approach (i.e. PILSMRK(L,U)).

$s = 2,\ k = 2:$

$$c^T = \begin{bmatrix} 0.39038820320221 & 1.00000000000000 \end{bmatrix}$$

$$G = \begin{bmatrix} -0.04671554852736 & 1.04671554852736 \\ -0.02010509586877 & 1.02010509586877 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.40044075113659 & -0.05676809646175 \\ 0.77072385847003 & 0.20917104566120 \end{bmatrix}$$

Crout:

$$\delta^T = \begin{bmatrix} 0.40044075113659 & 0.31843196932797 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1.00000000000000 & 0 \\ 9.39806495685529 & 1.00000000000000 \end{bmatrix}$$

Schur-Crout:

$$\delta^T = \begin{bmatrix} 0.36028586267747 & 0.35392212182843 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.06418485435680 & 0.05604152383747 \\ 0.99793802636797 & 0.99842843890084 \end{bmatrix}$$

$s = 2,\ k = 3:$

$$c^T = \begin{bmatrix} 0.42408624230810 & 1.00000000000000 \end{bmatrix}$$

$$G = \begin{bmatrix} 0.01290709720739 & -0.10843463813621 & 1.09552754092881 \\ 0.00354588047065 & -0.04623386039657 & 1.04268797992593 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.38745055226697 & -0.04598475368028 \\ 0.77239469511979 & 0.18846320542493 \end{bmatrix}$$

Crout:

$$\delta^T = \begin{bmatrix} 0.38745055226697 & 0.28013523838816 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1.00000000000000 & 0 \\ 7.19743219492460 & 1.00000000000000 \end{bmatrix}$$

Schur-Crout:

$$\delta^T = \begin{bmatrix} 0.33129449207677 & 0.32761955124138 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.08083975113162 & 0.07616492879483 \\ 0.99672711141866 & 0.99709523297510 \end{bmatrix}$$

$s = 4,\ k = 2:$

$$c^T = \begin{bmatrix} 0.09878664634426 & 0.43388702543882 & 0.80169299888049 & 1.00000000000000 \end{bmatrix}$$

$$G = \begin{bmatrix} -0.00087353889029 & 1.00087353889029 \\ 0.00062121019919 & 0.99937878980081 \\ -0.00032939714868 & 1.00032939714868 \\ -0.00003663563426 & 1.00003663563426 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.11996670457577 & -0.03384322082318 & 0.01835753398261 & -0.00656791028123 \\ 0.26010642038045 & 0.20159324902943 & -0.03956525951247 & 0.01237382574059 \\ 0.23561500946812 & 0.41088455735437 & 0.17597260265111 & -0.02110856774179 \\ 0.24141835002666 & 0.38984924120599 & 0.31101721961059 & 0.05767855352250 \end{bmatrix}$$

Crout:

$$\delta^T = \begin{bmatrix} 0.10617138884400 & 0.27770096849016 & 0.27497060030028 & 0.11996670457577 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0.01481904140434 \\ 0 & 0 & 0.00220678551539 & -0.02486729636785 \\ 0 & 0.38896861370956 & -0.38581432071631 & 0.05312025222596 \\ 1.00000000000000 & 0.92125100681023 & -0.92257381278026 & 0.99816844890362 \end{bmatrix}$$

Schur-Crout:

$$\delta^T = \begin{bmatrix} 0.17879165196884 & 0.15567835604316 & 0.18725864804630 & 0.18660124038403 \end{bmatrix}$$

$$Q = \begin{bmatrix} -0.05047735457027 & -0.05698986733483 & 0.04780729735701 & 0.04801402184956 \\ 0.17096598389037 & 0.17933373843615 & -0.16592112501907 & -0.16634392504346 \\ -0.15139062605533 & -0.08731023751861 & 0.17251778528806 & 0.17091936180350 \\ -0.97226722014607 & -0.97824766174222 & 0.96975370911955 & 0.96995408348419 \end{bmatrix}$$

$s = 4,\ k = 3:$

$$c^T = \begin{bmatrix} 0.10504182884419 & 0.44825417107884 & 0.80977028814179 & 1.00000000000000 \end{bmatrix}$$

$$G = \begin{bmatrix} 0.00007487445528 & -0.00195646912651 & 1.00188159467123 \\ -0.00007345206497 & 0.00148038414152 & 0.99859306792346 \\ 0.00003966973124 & -0.00083011136249 & 1.00079044163125 \\ 0.00000077039880 & -0.00008665832447 & 1.00008588792568 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.12388725564952 & -0.03052720746880 & 0.01502960651127 & -0.00515454606376 \\ 0.27600575210564 & 0.19832624728391 & -0.03534802573852 & 0.01060367743938 \\ 0.24659262259186 & 0.41336961213203 & 0.16850574024079 & -0.01944845872291 \\ 0.25397302181219 & 0.39037260118042 & 0.30064393200968 & 0.05492532747083 \end{bmatrix}$$

Crout:

$$\delta^T = \begin{bmatrix} 0.09980104557325 & 0.26112476902731 & 0.26633715617793 & 0.12388725564952 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0.01885332656568 \\ 0 & 0 & 0.00429974732457 & -0.03652952061848 \\ 0 & 0.38485479574542 & 0.39111661588660 & 0.09232717942550 \\ 1.00000000000000 & 0.92297713199827 & 0.92033108442036 & 0.99487981090186 \end{bmatrix}$$

Schur-Crout:

$$\delta^T = \begin{bmatrix} 0.17281106755693 & 0.15348751145786 & 0.18030166423062 & 0.17980311756297 \end{bmatrix}$$

$$Q = \begin{bmatrix} -0.03747602767829 & -0.04253832729434 & 0.03538720965186 & 0.03552524964325 \\ 0.15438230915055 & 0.16281308949598 & -0.14969201305536 & -0.15002487334226 \\ -0.16907928673314 & -0.11582715575719 & 0.18786790085145 & 0.18664755906307 \\ -0.97271467798559 & -0.97891085323893 & 0.97007509938672 & 0.97025418458887 \end{bmatrix}$$

References

[BHB92] Peter N. Brown, Alan C. Hindmarsh, and George D. Byrne. *VODE: A variable coefficient ODE solver*, August 1992. Available via WWW at URL http://www.netlib.org/ode/vode.f.

[Bin85] Zhou Bing. A-stable and L-stable block implicit one-block methods. *Journal of Computational Mathematics*, 3(4):328–341, 1985.

[But76] J. C. Butcher. On the implementation of implicit Runge–Kutta methods. *BIT*, 16:237–240, 1976.

[CGG$^+$91] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, New York, 1991.

[Cra94a] Cray Research, Inc. *CF77 Commands and Directives*, SR-3771 6.0 edition, 1994.

[Cra94b] Cray Research, Inc. *UNICOS Performance Utilities Reference Manual*, SR-2040 8.0 edition, 1994.

[GS69] A. Guillou and J. L. Soulé. La résolution numérique des problèmes différentiels aux conditions initiales par des méthodes de collocation. *R.I.R.O.*, R-3:17–44, 1969.

[HM96] P. J. van der Houwen and E. Messina. Parallel linear system solvers for Runge–Kutta–Nyström methods. Technical Report NM-R9613, CWI, Amsterdam, 1996. Submitted for publication.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer-Verlag, second revised edition, 1993.

[Hor76] E. H. Horneber. *Analyse nichtlinearer RLCÜ-Netzwerke mit Hilfe der gemischten Potentialfunktion mit einer systematischen Darstellung der Analyse nichtlinearer dynamischer Netzwerke.* PhD thesis, Universität Kaiserslautern, 1976.

[HS95] P. J. van der Houwen and J. J. B. de Swart. Triangularly implicit iteration methods for ODE-IVP solvers. Technical Report NM-R9510, CWI, Amsterdam, 1995. To appear in: SIAM Journal on Scientific Computing, 18(1), January 1997.

[HS96] P. J. van der Houwen and J. J. B. de Swart. Parallel linear system solvers for Runge–Kutta methods. Technical Report NM-R9616, CWI, Amsterdam, 1996. To appear in: Advances in Computational Mathematics.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[HW95] E. Hairer and G. Wanner. *RADAU5*, September 1995. Available via WWW at URL ftp://ftp.unige.ch/pub/doc/math/stiff/radau5.f.

[LN89] I. Lie and S. P. Nørsett. The stability function for multistep collocation methods. *Numer. Math.*, 57:779–787, 1989.

[LSV96] W. M. Lioen, J. J. B. de Swart, and W. A. van der Veen. Test set for IVP solvers. Report NM-R9615, CWI, Amsterdam, 1996. WWW version available at URL http://www.cwi.nl/cwi/projects/IVPtestset.shtml.

[Nev85] O. Nevanlinna. Matrix valued versions of a result of Von Neumann with an application to time discretization. *J. Comput. Appl. Math.*, 12 & 13:475–489, 1985.

[Pet91] L. R. Petzold. *DASSL: A Differential/Algebraic System Solver*, June 1991. Available via WWW at URL http://www.netlib.org/ode/ddassl.f.

[Sch94] S. Schneider. *Intégration de systèmes d'équations différentielles raides et différentielles-algébriques par des méthodes de collocations et méthodes générales linéaires.* PhD thesis, Université de Genéve, 1994.