Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Triangularly implicit iteration methods for ODE-IVP solvers

P.J. van der Houwen and J.J.B. de Swart

Department of Numerical Mathematics

NM-R9510 1995

Report NM-R9510
ISSN 0169-0388

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

Triangularly Implicit Iteration Methods for ODE-IVP Solvers

P.J. van der Houwen & J.J.B. de Swart
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

It often happens that iteration processes used for solving the implicit relations arising in ODE-IVP methods only start to converge rapidly after a certain number of iterations. Fast convergence right from the beginning is particularly important if we want to use so-called step-parallel iteration in which the iteration method is concurrently applied at a number of step points. In this paper, we construct highly parallel iteration methods that do converge fast from the first iteration on. Our starting point is the PDIRK method (parallel, diagonal-implicit, iterated Runge-Kutta method), designed for solving implicit Runge-Kutta equations on parallel computers. The PDIRK method may be considered as Newton type iteration in which the Newton Jacobian is 'simplified' to block-diagonal form. However, when applied in a step-parallel mode, it turns out that its relatively slow convergence, or even divergent behaviour, reduces the effectiveness of the step-parallel scheme. By replacing the block-diagonal Newton Jacobian approximation in PDIRK by a block-triangular approximation, we do achieve convergence right from the beginning at a modest increase of the computational costs. Our convergence analysis of the block-triangular approach will be given for the wide class of general linear methods, but the derivation of iteration schemes is limited to Runge-Kutta based methods. A number of experiments show that the new parallel, triangular-implicit, iterated Runge-Kutta method (PTIRK method) is a considerable improvement over the PDIRK method.

CR Subject Classification (1991): G.1.7
Keywords and Phrases: numerical analysis, convergence of iteration methods, Runge-Kutta methods, parallelism.

Note: The research reported in this paper was partly supported by the Technology Foundation (STW) in the Netherlands.

1. Introduction

Suppose that we integrate the IVP

$$(1.1) \qquad \frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}, \mathbf{f} \in R^d$$

by an implicit step-by-step method. For the class of general linear methods (cf. Butcher [7, p.367]), this requires in each step the solution of a nonlinear system of the form

$$(1.2) \qquad \mathbf{R}(\mathbf{Y}) = \mathbf{0}, \quad \mathbf{R}(\mathbf{Y}) := \mathbf{Y} - h(A\otimes I)\mathbf{F}(\mathbf{Y}) - \mathbf{W},$$

where A denotes a nonsingular s-by-s matrix, **W** is an sd-dimensional vector containing information computed in preceding integration steps, I is the d-by-d identity matrix, h is the stepsize $t_n - t_{n-1}$, and $\otimes$ denotes the Kronecker product. The s components $\mathbf{Y}_i$ of the sd-dimensional solution vector **Y** represent s numerical approximations to the s exact solution vectors $\mathbf{y}(\mathbf{e}t_{n-1} + \mathbf{c}h)$; here, **c** denotes the abscissa vector and **e** is the vector with unit entries. Furthermore, for any vector $\mathbf{V} = (\mathbf{V}_i)$, $\mathbf{F}(\mathbf{V})$ contains the derivative values $(\mathbf{f}(\mathbf{V}_i))$. It is assumed that the components of **c** are distinct. In the following, we shall use the notation I for any identity matrix. However, its order will always be clear from the context.

The solution **Y** of (1.2) will be called the stage vector and s will be referred to as the number of stages. The most well-known examples of step-by-step methods that leads to implicit relations of the form (1.2) are provided by the class of implicit Runge-Kutta (RK) methods. In that case, s equals the number of *implicit* stages of the RK method (note that for RK methods having *explicit* stages, s is less than the *total* number of stages of the RK method, e.g. this happens for Lobatto methods).

The system (1.2), to be referred to as the corrector, will be solved iteratively by generating a sequence of iterates $\{\mathbf{Y}^{(j)}\}$. Unfortunately, it often happens that iteration processes for solving (1.2) only start to converge rapidly after a certain number of iterations. In particular, iteration methods whose error amplification matrix is nonnormal often exhibit such a behaviour, in spite of the fact that the eigenvalues of the amplification matrix are of small magnitude. Fast convergence right from the beginning (rather than fast convergence in subsequent iterations) is particularly important if we want to use so-called *step-parallel* iteration methods. In such methods, the iteration procedure is concurrently applied at a number of step points, that is, iteration at the point t_{n+1} is already started without waiting until the iterates $\mathbf{Y}^{(j)}$ at t_n have converged. This approach requires that the predictor formula needed to start iteration at t_n is based on a sufficiently "safe" iterate $\mathbf{Y}^{(j)}$. In order to have an efficient step-parallel iteration process, the value of j for which $\mathbf{Y}^{(j)}$ is sufficiently "safe" should be small, that is, substantially smaller than the order of the method (1.2). Step-parallel methods and its various versions have been discussed and analysed in a number of papers, among which Miranker and Liniger [19], Bellen [1], Bellen et al. [2,3], Burrage [4,5], Gear and Xu Xuhai [11], Chartier [8], and Van der Houwen et al. [16,17].

In the step-parallel approach in [16,17] we used RK methods as the underlying corrector and the PDIRK method (Parallel, Diagonal-implicit Iterated Runge-Kutta method) as the underlying iteration scheme [15]. The PDIRK method may be considered as Newton type iteration in which the Newton Jacobian is 'simplified' to block-diagonal form. This iteration method is quite efficient when iterating until convergence, but it does have the drawback of a rather slow initial convergence. The aim of the present paper is to improve the PDIRK approach such that we have convergence right from the beginning. On the basis of a linear convergence theory, we have designed an iteration strategy that does yield relatively fast initial convergence for a number of classical RK correctors, at a modest increase of the computational costs. We tested this strategy on a few nonlinear problems from the literature. These experiments do show a considerable improvement of the initial speed of convergence. Since in step-parallel methods, fast initial convergence is crucial, the strategy designed

in this paper should nicely fit into a step-parallel iteration approach. This will subject of future research.

2. The iteration scheme

Our starting point for solving the corrector equation (1.2) is the simplified (or modified) Newton iteration scheme

$$(2.1) \qquad \left(I - A\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)} = -\,\mathbf{R}\left(\mathbf{Y}^{(j)}\right),\quad \mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)},\qquad j = 0, 1, \ldots ,$$

where J is an approximation to the Jacobian of the righthand side function **f** at t_{n-1}, and $\mathbf{Y}^{(0)}$ is the initial iterate to be provided by some predictor formula. Each iteration with (2.1) requires the solution of an sd-dimensional linear system for the Newton correction $\Delta\mathbf{Y}^{(j+1)}$. In actual computation, the costs for solving this system can be reduced by first performing a similarity transformation of the iterates (cf. Butcher [6]), $\mathbf{Y}^{(j)} = (Q\otimes I)\mathbf{X}^{(j)}$, where Q is a nonsingular matrix. Q should be such that the system

$$(2.1') \qquad \left(I - Q^{-1}AQ\otimes hJ\right)\Delta\mathbf{X}^{(j+1)} = -\,(Q^{-1}\otimes I)\,\mathbf{R}\left(\mathbf{Y}^{(j)}\right),\quad \mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + (Q\otimes I)\Delta\mathbf{X}^{(j+1)},\quad j = 0, 1, \ldots ,$$

is easier to solve than (2.1).

For example, if A has positive eigenvalues, then the Schur decomposition of A has the form $A = QTQ^{-1}$, where Q is unitary and T is lower triangular with the eigenvalues of A on its diagonal. Hence, the linear system (2.1') is 'triangularly implicit' and consists of s subsystems of dimension d that can be solved sequentially. On sequential computers, this is most effective if A has a one-point spectrum, so that only one LU-decomposition is required. On parallel computers, the condition on the spectrum of A can be relaxed to the requirement that A has arbitrarily positive eigenvalues. Since the s LU-decompositions can be computed in parallel, effectively only one decomposition per processor is required. Similarly, the s components of the residue $\mathbf{R}\left(\mathbf{Y}^{(j)}\right)$ can also be computed in parallel. Furthermore, we have to perform s forward-backward substitutions in each iteration. If we impose the additional requirement that in (2.1') the matrix $T = Q^{-1}AQ$ is diagonal, then the s subsystems are uncoupled and in each iteration the forward-backward substitutions can also be done concurrently. RK methods whose RK matrices have *real* eigenvalues can be found in Orel [20].

Unfortunately, the most powerful implicit methods have matrices A with *complex* eigenvalues. One option to deal with the complex eigenvalue case is to decompose $Q^{-1}AQ$ into a real block- triangular matrix of which the diagonal blocks are either diagonal or 2-by-2 matrices. This leads to an sd-dimensional system that can be split into a sequence of subsystems either of dimension d or of dimension 2d (this approach was followed in the implementation of the RADAU5 code of Hairer and

Wanner [14]). A block-diagonal structure of $Q^{-1}AQ$ implies that (2.1') is suitable for implementation on a parallel system.

An other option for reducing computational costs, that will be the subject of this paper, replaces the matrix A in (2.1) by a 'more convenient' matrix B. In this paper, we consider the case where B is lower triangular, i.e. B = L + D, where L is strictly lower triangular and D is diagonal with positive diagonal entries d_{ii}. This leads to the iteration scheme

$$(2.2)\qquad \left(I - D\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)} = \left(L\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)} - \mathbf{R}(\mathbf{Y}^{(j)}),\quad \mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)},\qquad j = 0, 1, \ldots .$$

In the case where L vanishes and (1.2) represents a Runge-Kutta (RK) method, the resulting iteration scheme is the PDIRK method mentioned in Section 1. The method (2.2) requires LU-decompositions of the d-by-d matrices $I - hd_{ii}J$, i = 1, ... , s, and, in each iteration, the evaluation of the residue $\mathbf{R}(\mathbf{Y}^{(j)})$, s forward-backward substitutions, and the matrix-vector multiplication $\left(L\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)}$. By expressing this multiplication in terms of the righthand side function **F**, the scheme (2.2) can be replaced by

$$(2.3)\qquad \left(I - D\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)} = h\left(L\otimes I\right)\left(\mathbf{F}(\mathbf{Y}^{(j+1)}) - \mathbf{F}(\mathbf{Y}^{(j)})\right) - \mathbf{R}(\mathbf{Y}^{(j)}).$$

This version may yield better convergence if the righthand side Jacobian is a less accurate approximation to the true Jacobian. Just like the scheme (2.1'), the LU-decompositions and the components of the residue $\mathbf{R}(\mathbf{Y}^{(j)})$ occurring in (2.2) and (2.3) can be evaluated in parallel. The schemes (2.2) and (2.3) will be called *parallel, triangular-implicit, iterated* methods. In the case where (1.2) is an RK method, we shall refer to such methods as a PTIRK method and to distinguish them, we shall speak of the *LJ* and *LF version*.

In the case of (2.2), a further degree of parallelism is obtained by using the Butcher similarity transformation. This enables us to get rid of the triangular-implicit term $\left(L\otimes hJ\right)\Delta\mathbf{Y}^{(j+1)}$ and leads to

$$(2.4)\qquad \left(I - D\otimes hJ\right)\Delta\mathbf{X}^{(j+1)} = -\left(Q^{-1}\otimes I\right)\mathbf{R}(\mathbf{Y}^{(j)}),\quad \mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + (Q\otimes I)\Delta\mathbf{X}^{(j+1)},\quad BQ = QD.$$

In addition to the parallelism already present in (2.2) and (2.3), the scheme (2.4) also allows that in each iteration the s forward-backward substitutions can be done in parallel. Since the schemes (2.2) and (2.4) are algebraicly identical,we shall call (2.4) the *transformed LJ version*.

Finally, we compare the computational costs of the various iteration schemes. These costs consists of two contributions, respectively due to Jacobian updates and due to the successive iterations. In all schemes, the number of flops per step originating from the Jacobian updates is given by

$$C_1 = \frac{sd^2}{\nu}\left(\frac{1}{s}C_J + \frac{2}{3}d + 1\right),$$

where ν denotes the averaged number of steps during which the Jacobian and the LU-decomposition is kept constant, and C_J denotes the average numbers of flops for computing one entry of J. The contribution C_1 is perfectly parallelizable and effectively, can be reducd by a factor s on s processors. The contribution due to m (say) iterations are summarized in Table 2.1 (for a specification of these costs, we refer to Appendix B of this paper). In this table, C_f denotes the average numbers of flops for computing one component of **f**. Evidently, on a parallel computer, the methods (2.2) with L = O and (2.4) are the less expensive ones.

Table 2.1. Computational costs due to m iterations.

Method	on one processor	on s processors
(2.2) with L = O	$msd(C_f + 2d + 2s)$	$md(C_f + 2d + 2s)$
(2.2) with L ≠ O: LJ version	$msd(C_f + 4d + 3s)$	$md(C_f + 4ds + s^2)$
(2.3) with L ≠ O: LF version	$msd(C_f + 2d + 3s)$	$md(sC_f + 2ds + s^2)$
(2.4): transformed LJ version	$msd(C_f + 2d + 4s)$	$md(C_f + 2d + 4s)$

3. Convergence of the iteration process

In order to analyse convergence, we define the iteration error $\varepsilon^{(j)} = \mathbf{Y}^{(j)} - \mathbf{Y}$, and we write the LJ and LF versions (2.2) and (2.3) in the respective forms

$$(3.1)\qquad \left(I - B\otimes hJ\right)\left(\varepsilon^{(j+1)} - \varepsilon^{(j)}\right) = -\varepsilon^{(j)} + h\left(A\otimes I\right)\left(\mathbf{F}(\mathbf{Y} + \varepsilon^{(j)}) - \mathbf{F}(\mathbf{Y})\right),$$

$$(3.2)\qquad \left(I - D\otimes hJ\right)\left(\varepsilon^{(j+1)} - \varepsilon^{(j)}\right) = -\varepsilon^{(j)} + h\left((A - L)\otimes I\right)\left(\mathbf{F}(\mathbf{Y} + \varepsilon^{(j)}) - \mathbf{F}(\mathbf{Y})\right)$$
$$+\ h(L\otimes I)\left(\mathbf{F}(\mathbf{Y} + \varepsilon^{(j+1)}) - \mathbf{F}(\mathbf{Y})\right).$$

The components of $\mathbf{F}(\mathbf{Y} + \varepsilon) - \mathbf{F}(\mathbf{Y})$ can be expanded according to $J_i\,\varepsilon_i + O(\varepsilon_i^2)$, where J_i is the Jacobian matrix of the righthand side function at $\mathbf{Y}_i$. Assuming that J is nonsingular, we may define the block-diagonal matrix ΔJ of which the diagonal blocks are given by $J^{-1}\Delta J_i = J^{-1}(J_i - J)$ to obtain

$$(3.3)\qquad \mathbf{F}(\mathbf{Y} + \varepsilon^{(j)}) - \mathbf{F}(\mathbf{Y}) = (I\otimes J)\,\varepsilon^{(j)} + (I\otimes J)\Delta J\,\varepsilon^{(j)} + O\left((\varepsilon^{(j)})^2\right).$$

Ignoring the second-order terms (first-order convergence analysis), the error recursions for the LJ and LF versions can be represented in the forms

$$(3.4)\qquad \varepsilon^{(j+1)} = M\left(I + P\Delta J\right)\varepsilon^{(j)},$$

$$(3.5)\qquad \varepsilon^{(j+1)} = \left(I - N\Delta J\right)^{-1} M\left(I + Q\Delta J\right)\varepsilon^{(j)},$$

where

(3.6)
$$M := \left(I - B\otimes hJ\right)^{-1}\left((A - B)\otimes hJ\right),$$
$$N := \left(I - B\otimes hJ\right)^{-1}\left(L\otimes hJ\right),\ P = (A - B)^{-1}A\otimes I,\quad Q = (A - B)^{-1}(A-L)\otimes I.$$

If we ignore ΔJ (linear convergence analysis), then the error recursions of both versions are characterized by the matrix M. However, if ΔJ cannot be neglected, then the error recursions may behave quite differently. For example, as $h \to 0$, then we have

(3.4') $$\varepsilon^{(j+1)} \approx h\left((A - B)\otimes J + (A\otimes J)\Delta J\right)\varepsilon^{(j)},$$

(3.5') $$\varepsilon^{(j+1)} \approx h\left((A - B)\otimes J + ((A-L)\otimes J)\Delta J\right)\varepsilon^{(j)}.$$

Since the strictly lower triangular blocks of the amplification matrices in (3.4') and (3.5') differ by the matrices $hL_{ij}\Delta J_j$, the convergence behaviour may differ considerably and is highly problem dependent. In the remainder of this paper, we shall focus on the matrix M.

3.1. Rate of convergence

In order to select a suitable matrix B, we consider the convergence of the individual error components corresponding to the eigenvalues λ of J. From (3.1) it follows that these error components are amplified by the matrix Z defined by

(3.7) $$Z = Z(z) = z\left(I - zB\right)^{-1}\left(A - B\right),\qquad z := h\lambda.$$

Z will be called the *amplification matrix associated with M*.
A measure for the rate of convergence of the individual error components is defined by

(3.8) $$\rho_j(z) := \sqrt[j]{\| \left(Z(z)\right)^j \|_\infty}\ .$$

where $\| \|_\infty$ denotes the maximum norm. Note that $\rho_\infty(z) = \rho\left(Z(z)\right)$, $\rho(.)$ being the spectral radius function. For the test equation $y' = \lambda y$, the value of $\rho_j(z)$ may be interpreted as the averaged factor by which the iteration error corresponding to $z = h\lambda$ is reduced in each iteration, until the corrector solution is reached. For more general problems, we have to deal with $\rho_j(z)$ where z runs through the spectrum of hJ.
The convergence rate at infinity will be called the *stiff* rate of convergence. In the neighbourhood of the origin we may write

(3.9) $\rho_j(z) = |z| \sqrt[j]{|| (A - B)^j ||} + O(z^2) =: \tilde{\rho}_j(z) + O(z^2)$ as $z \to 0$.

The quantity $\tilde{\rho}_j(z)$ will be called the *nonstiff* rate of convergence. Furthermore, we denote the maximal rate of convergence in the lefthand plane by ρ_j^*. Of course, ρ_j^* refers to the worst case situation, but it serves as an indicator for the robustness of the method.

3.2. Iteration strategies

In this section, we discuss the choice of the free matrix $B = L + D$ in the iteration schemes (2.2) and (2.3). We first briefly review the *diagonal* iteration strategy of [15], i.e. $L = O$, and then we focus on the *triangular* iteration strategy where L is allowed to be an arbitrary strictly (lower) *triangular* matrix. A nonvanishing matrix L enables us to reduce the norm of Z^j considerably.

The reason for restricting B to the class of triangular matrices is that we have direct control on the eigenvalues of B. As a consequence, suitable matrices B can be constructed without performing a many-parameter search as was carried out in [15]. Since our main source of correctors is the class of RK methods which usually possess a dominant *lower* triangular part, B is also assumed to be *lower* triangular (recall that ideally B should equal A). Both for the diagonal iteration and the triangular iteration approach, the matrices B, $Z_0 := A - B$ and $Z_\infty := I - B^{-1}A$ associated with a number of classical RK methods are specified in the Appendix to this paper.

3.2.1. Diagonal iteration. The diagonal iteration strategy is characterized by a diagonal matrix B with positive diagonal entries. In this strategy, it was found for a large number of classical RK correctors that a small stiff rate of convergence is crucial for a satisfactory overall convergence [15]. Therefore, in [15] the diagonal matrix $B = D$ was obtained by a multi-parameter search such that Z_∞ has a minimal spectral radius, that is, the *asymptotic* value $\rho_\infty(\infty)$ of the stiff rate of convergence is minimized. For a large number of collocation based RK correctors, it turned out that the spectral radius $\rho(Z_\infty)$ of $Z_\infty = I - B^{-1}A$ is extremely small. In fact, we conjecture that for collocation based RK correctors, there exist matrices D with positive diagonal entries for which $\rho(Z_\infty)$ actually vanishes. This suggests an alternative construction of the matrix $B = D$. Writing down the characteristic equation for Z_∞ and imposing the condition that this equation has only zero roots, we arrive at a (nonlinear) system for the entries d_{ii} of D. If this system can be solved for positive d_{ii}, $i = 1, \ldots, s$, then we have found an optimal matrix D. It has been verified for the Radau IIA correctors with $s \le 8$ that such optimal matrices D do exist (see [18]). Notice that a zero spectral radius $\rho(Z_\infty)$ implies that Z_∞^j vanishes for $j \ge s$ (this can be seen by considering the Schur decomposition $Z_\infty = QTQ^{-1}$ with Q unitary and T strictly lower triangular).

If the matrices D are obtained by a numerical search as in [15], then they will always give rise to a small but yet nonzero $\rho(Z_\infty)$. Nevertheless, both for the *nonstiff* and the *highly stiff* error components, the generated PDIRK methods show a satisfactory convergence rate for larger values of j. On the other hand, it also turns out that for the higher-order methods, there may be regions in the z-

plane where $\rho_j(z)$ exceeds one for small j, so that initially error components corresponding to points lying in such regions will diverge [17, Tables 3.2b]. The reason for this behaviour is the 'abnormality' of the matrix Z. In particular, for larger values of |z|, i.e. for the stiff error components, the matrix Z(z) may differ considerably from a normal matrix. To be more precise, let the departure from normality of the matrix Z be defined by $\Delta^2(Z) := \|Z\|_F^2 - \|\zeta(Z)\|_2^2$, where $\zeta(Z)$ denotes the vector of eigenvalues of Z and $\| \|_F$ and $\| \|_2$ respectively denote the Frobenius matrix norm and the Euclidean vector norm (see e.g. [12, p.194]). By considering plots of $\Delta^2(Z)$ as a function of |z| with arg(z) constant, we found that in the lefthand halfplane $\Delta^2(Z)$ monotonically increases from 0 to values greater than 20. This situation is particularly unfortunate if we want to apply the step-parallel iteration approach mentioned in Section 1. In such an approach, it is crucial that in the whole lefthand halfplane the convergence rate is less than one right from the beginning.

3.2.2. Triangular iteration. The triangular iteration strategy is characterized by a lower triangular B with positive diagonal entries such that Z_∞ is *strictly upper triangular*. As a consequence, we have a zero stiff rate of convergence for $j \geq s$. The matrix Z_∞ can be constructed by using the LU decomposition of A. Let $A = T_L T_U$ with T_L lower triangular and T_U unit upper triangular (Crout decomposition), and define $B = T_L$. Since $Z_\infty = I - B^{-1}A$, we immediately obtain the strictly upper triangular matrix $Z_\infty = I - T_U$. The following lemma provides an explicit criterion for the positiveness of the diagonal entries of $B = T_L$.

Lemma 3.1. Let A, L, D, and U be s-by-s matrices such that A = LDU with L unit lower triangular, D diagonal and U unit upper triangular, and let A_k denote the k-by-k principal submatrix of A. Then D has diagonal entries given by

$$(3.10) \quad d_k = \frac{\det (A_k)}{\det (A_{k-1})} \ , \ k = 1, \ldots , s,$$

where $\det (A_0) := 1$ and $\det (A_1) := a_{11}$.

Proof. Let A_i be decomposed according to $A_k = L_k D_k U_k$ with L_k unit lower triangular, D_k diagonal and U_k unit upper triangular. Then $\det (A_k) = \det (L_k) \det (D_k) \det (U_k) = \det (D_k)$. Since the first k-1 pivots in the Gaussian elimination process do not depend on the entries a_{ij} with $i \geq k$ and $j \geq k$, it follows that the diagonal entries d_{ik} of D_k are defined by $d_{ik} = d_i$. Hence, $\det (D_k) = \det (D_{k-1}) d_k$, which is equivalent with (3.10). **[]**

From this lemma it follows that the diagonal entries of the matrix B defined above are given by (3.10), so that they are all positive, if all values $\det (A_k)$, k = 1, ... , s, are positive.
In the following, we restrict our considerations to collocation methods with distinct abscissas c_i. Such methods are generated by matrices A of the form

(3.11) $A = CVRV^{-1}$, $C = \text{diag}(\mathbf{c})$, $R = \text{diag}(\mathbf{r})$, $\mathbf{c} = (c_i)$, $\mathbf{r} = (i^{-1})$, $V = \left(\mathbf{e}\ \mathbf{c}\ \mathbf{c}^2 \ldots \mathbf{c}^{s-1}\right)$,

where $i = 1, \ldots, s$.

Theorem 3.1. If A results from a collocation method with positive, distinct abscissas ordered according to $0 < c_1 < c_2 < \ldots < c_s$, then the following results hold:

(a) The values of $\det(A_1)$ and $\det(A_s)$ are positive for all s.

(b) Let V and R be partitioned according to (3.12), where V_k and R_k denote the k-by-k principal submatrices of the matrices V and R defined in (3.11). Then, for $1 < k < s$, the value of $\det(A_k)$ is positive if

(3.12) $$q_k := \det(V_k R_k - PSW^{-1}Q) > 0, \quad V = \begin{pmatrix} V_k & P \\ Q & W \end{pmatrix}, \quad R = \begin{pmatrix} R_k & O \\ O & S \end{pmatrix}.$$

Proof. (a) For collocation methods we have that

(3.13) $$a_{11} = \int_0^{c_1} \frac{c_2 - t}{c_2 - c_1} \frac{c_3 - t}{c_3 - c_1} \cdots \frac{c_s - t}{c_s - c_1}\, dt.$$

From the condition on the collocation points it is immediate that $\det(A_1) = a_{11} > 0$ and from (3.11) it follows that $\det(A) = \det(C)\det(VRV^{-1}) = \det(C)\det(R) > 0$.

(b) By means of (3.11) it is easily verified that A_k can be presented in the form

(3.14) $$A_k = C_k(V_k R_k - PSW^{-1}Q)(V_k - PW^{-1}Q)^{-1},$$

where C_k is the k-by-k principal submatrix of C and V_k, R_k, P, S, W and Q are specified in (3.12). We now prove that $\det(V_k - PW^{-1}Q)$ is positive by considering the factorizations

$$\begin{pmatrix} V_k - PW^{-1}Q & O \\ Q & W \end{pmatrix} = \begin{pmatrix} I & -PW^{-1} \\ O & I \end{pmatrix} V$$

and

$$\begin{pmatrix} V_k - PW^{-1}Q & O \\ Q & W \end{pmatrix} = \begin{pmatrix} V_k - PW^{-1}Q & O \\ O & I \end{pmatrix} \begin{pmatrix} I & O \\ Q & W \end{pmatrix}.$$

From these two relations it follows that $\det(V) = \det(V_k - PW^{-1}Q)\det(W)$. Since V is a VanderMonde matrix and W is a row-scaled VanderMonde matrix, we conclude that both V and W have a positive determinant. Thus, $\det(V_k - PW^{-1}Q) > 0$, and by virtue of (3.14), it follows that $\det(A_k)$ is positive whenever the quantity q_k defined in (3.12) is positive. **[]**

Theorem 3.1 directly implies the positiveness of the diagonal entries of $B = T_L$ for all two-stage collocation methods. For higher-stage methods, it provides the relatively simple criterion $q_k > 0$, $1 < k < s$, for verifying the condition $\det(A_k) > 0$. We conjecture that the condition $\det(A_k) > 0$, $1 \le k \le s$, is true for all s, but so far we are not able to prove it. Instead, we verified the correctness of this conjecture for $s \le 6$. An easy way of verifying the conditions $q_k > 0$, replaces the abscissas c_i in q_k by $c_i = p_i + p_{i-1} + ... + p_1$ and expresses q_k as a rational function of the s parameters p_i. For $s \le 6$, it turns out that all coefficients in this rational expression are positive. Since the parameters p_i are all positive (because $p_i := c_i - c_{i-1}$ with $c_0 := 0$), this implies that q_k is positive.

Example 3.1. For s = 3, we have to prove that $q_2 > 0$. A straightforward calculation yields

$$q_2 = \frac{3p_2p_3^2 + 4p_2^2p_3 + 2p_1p_2p_3 + p_1p_2^2 + p_2^3}{6(p_1 + p_2 + p_3)^2} \ ,$$

which is obviously positive. **[]**

Summarizing we conclude that, unlike the diagonal approach, the triangular approach provides an extremely simple construction of the matrix B and an explicit criterion for checking the positiveness of its diagonal entries. Moreover, it turns out that for larger values of |z| the the departure from normality $\Delta^2(Z)$ defined in the preceding subsection is considerably reduced. Plots show that $\Delta^2(Z)$ monotonically increases from 0 at the origin to values less than than 0.4 at infinity, resulting in rates of convergence that are less than one in the whole left halfplane for all j. This will be quantified in the following subsection.

3.3. Comparison of convergence rates

For a number of well-known RK correctors, we compare the convergence rates defined by (3.8) associated with the diagonal approach (PDIRK method) and the triangular approach (PTIRK method). For j = 1, 2, 3, Table 3.1 presents the *nonstiff* rate of convergence $\tilde{\rho}_j(z)$ as defined in (3.9), the *stiff* rate of convergence $\rho_j(\infty)$, and the maximal rate of convergence ρ_j^*. These figures indicate that in the first few iterations, the PTIRK strategy converges considerably faster than the PDIRK strategy. Hence, it should be a sound starting point for step-parallel applications. This will be subject of future research.

Table 3.1. Convergence rates $\tilde{\rho}_j$, $\rho_j(\infty)$ and ρ^*_j.

Corrector	s	Method	$\tilde{\rho}_1(z)$	$\tilde{\rho}_2(z)$	$\tilde{\rho}_3(z)$	$\rho_1(\infty)$	$\rho_2(\infty)$	$\rho_3(\infty)$	ρ_1^*	ρ_2^*	ρ_3^*	ρ_∞^*
Gauss	2	PDIRK	0.79z	0.36z	0.45z	1.58	0	0	1.58	0.59	0.45	0.25
		PTIRK	0.08z	0.08z	0.08z	0.15	0	0	0.15	0.14	0.14	0.14
Radau IIA	2	PDIRK	1.15z	0.52z	0.40z	1.78	0	0	1.78	0.63	0.47	0.26
		PTIRK	0.15z	0.15z	0.15z	0.20	0	0	0.20	0.18	0.18	0.18
Lobatto IIIA	2	PDIRK	0.89z	0.31z	0.21z	2.29	0	0	2.29	0.58	0.39	0.17
		PTIRK	0.08z	0.08z	0.08z	0.13	0	0	0.13	0.14	0.14	0.14
Radau IIA	3	PDIRK	1.15z	0.44z	0.34z	4.17	1.82	0	4.17	1.82	1.02	0.40
		PTIRK	0.21z	0.20z	0.20z	0.46	0.26	0	0.46	0.40	0.39	0.37
Lobatto IIIA	3	PDIRK	0.91z	0.32z	0.30z	5.62	3.09	0	5.62	3.09	1.35	0.45
		PTIRK	0.13z	0.13z	0.12z	0.23	0.17	0	0.23	0.33	0.32	0.30
Radau IIA	4	PDIRK	1.10z	0.52z	0.25z	4.68	3.31	2.09	4.68	3.31	2.09	0.52
		PTIRK	0.25z	0.22z	0.20z	0.68	0.47	0.30	0.68	0.58	0.55	0.50

4. Numerical illustration

In this section, we compare the diagonal-implicit iteration strategy with the triangular-implicit iteration strategy. In all experiments, we used the 4-stage Radau IIA corrector with constant stepsizes (the initial condition in the problems below is adapted such that the integration starts outside the transient phase enabling us to use constant steps). Two predictors were tested, the simple last step value (LSV) predictor $\mathbf{Y}^{(0)} = (\mathbf{e}_s^T \otimes I)\mathbf{Y}$ and the extrapolation (EPL) predictor $\mathbf{Y}^{(0)} = (E \otimes I)\mathbf{Y}$. Here, $\mathbf{Y}$ denotes the stage vector from the preceding step, $\mathbf{e}_s$ is the sth unit vector, and E is the extrapolation matrix.

In the tables of results, the LF and LJ versions (2.2) and (2.3) of the PTIRK method are indicated by PTIRK(LJ) and PTIRK(LF). For a given number of iterations m, these tables present the minimal number of correct digits cd of the components of the numerical solution at the end point t = T of the integration interval (that is, at the end point the absolute errors are written as 10^{-cd}).

These tables clearly show for both predictors the superiority of the PTIRK strategy in the first few iterations. For large numbers of iterations, PDIRK and PTIRK(LJ) are better than PTIRK(LF).

Table 4.1a. LSV predictor: HIRES problem (4.1) of Schäfer.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	15	*	*	*	4.3	…	6.5
PTIRK(LJ)	15	3.4	3.5	3.8	4.2	…	6.3
PTIRK(LF)	15	3.1	4.0	3.9	4.1	…	5.6
PDIRK	7.5	*	*	*	5.4	…	7.7
PTIRK(LJ)	7.5	4.0	4.2	4.7	5.1	…	8.3
PTIRK(LF)	7.5	3.3	4.4	4.7	5.3	…	7.0

Table 4.1b. EPL predictor: HIRES problem (4.1) of Schäfer.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	15	*	*	*	*	…	6.4
PTIRK(LJ)	15	*	3.0	4.8	5.1	…	7.3
PDIRK	7.5	*	*	*	4.1	…	8.8
PTIRK(LJ)	7.5	*	2.5	6.1	6.6	…	9.0

4.1. HIRES problem of Schäfer

Our first example is provided by a problem of Schäfer (called the HIRES problem in [14, Vol. II p.157]). It was proposed in Gottwald [13] as a test problem and consists of 8 mildly stiff equations on the interval $5 \le t \le 305$:

$$
\begin{aligned}
y_1' &= -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007, && y_1(5) = 0.0316516757045,\\
y_2' &= +1.71y_1 - 8.75y_2, && y_2(5) = 0.0064815495310,\\
y_3' &= -10.03y_3 + 0.43y_4 + 0.035y_5, && y_3(5) = 0.0045834510647,\\
y_4' &= +8.32y_2 + 1.71y_3 - 1.12y_4, && y_4(5) = 0.0897432327351,\\
y_5' &= -1.745y_5 + 0.43y_7 + 0.43y_6, && y_5(5) = 0.1624514537526,\\
y_6' &= -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7, && y_6(5) = 0.6850438961444,\\
y_7' &= +280y_6y_8 - 1.81y_7, && y_7(5) = 0.0056467003419,\\
y_8' &= -280y_6y_8 + 1.81y_7, && y_8(5) = 0.0000532996581.
\end{aligned}
\tag{4.1}
$$

4.2. CHREAC problem of Gear

This problem is a set of chemical reaction equations originating from Gear [10] on the interval [1,51] and used in the test set of Enright et al. [9]:

$$
\begin{aligned}
y_1' &= -0.013y_1 - 1000y_1y_3, && y_1(1) = 0.990731920827,\\
y_2' &= -2500y_2y_3, && y_2(1) = 1.009264413846,\\
y_3' &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3, && y_3(1) = -0.366532612659\ 10^{-5}.
\end{aligned}
\tag{4.2}
$$

Table 4.2a. LSV predictor: CHREAC problem (4.2) of Gear.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	50	1.4	2.2	2.6	2.9	…	5.2
PTIRK(LJ)	50	2.3	2.7	3.5	4.3	…	7.7
PTIRK(LF)	50	1.8	2.9	3.9	3.0	…	3.3
PDIRK	25	1.8	2.9	3.4	3.6	…	7.3
PTIRK(LJ)	25	2.3	3.6	4.2	5.3	…	9.8
PTIRK(LF)	25	2.1	4.3	4.4	4.6	…	6.4

Table 4.2b. EPL predictor: CHREAC problem (4.2) of Gear.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	25	2.4	2.8	3.2	3.6	…	7.4
PTIRK(LJ)	25	2.9	3.7	4.3	5.6	…	9.8

4.3. ATMOS20 problem of Verwer

The ATMOS20 problem is a system of 20 stiff nonlinear ODEs originating from an air pollution model (see Verwer [21]). We solved this system in the integration interval [5,60].

Table 4.3a. LSV predictor: ATMOS20 problem of Verwer.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	11	2.7	2.1	2.8	5.4	..	9.8
PTIRK(LJ)	11	3.3	5.0	6.1	6.7	…	11.0
PTIRK(LF)	11	3.4	4.9	7.0	6.8	…	8.7
PDIRK	5.5	1.3	*	*	6.5	…	11.2
PTIRK(LJ)	5.5	3.7	5.6	7.0	7.7	…	12.2
PTIRK(LF)	5.5	3.7	5.5	7.6	8.3	…	11.5
PDIRK	2.25	*	*	*	7.5	…	12.2
PTIRK(LJ)	2.25	4.0	6.2	7.8	8.6	…	12.6
PTIRK(LF)	2.25	4.0	6.2	8.2	10.0	…	12.1

Table 4.3b. EPL predictor: ATMOS20 problem of Verwer.

Strategy	h	m=1	m=2	m=3	m=4	…	m=10
PDIRK	11	1.8	2.6	2.1	5.4	..	10.0
PTIRK(LJ)	11	2.0	3.7	6.3	7.0	…	10.9
PDIRK	5.5	*	*	*	6.4	…	11.8
PTIRK(LJ)	5.5	*	4.2	7.4	8.2	…	12.2
PDIRK	2.25	*	*	*	8.2	…	12.7
PTIRK(LJ)	2.25	*	*	8.6	9.4	…	12.7

Acknowledgement. The authors are grateful to Dr. B.P. Sommeijer and to Dr. W. Hoffmann for the many valuable discussions on this paper.

References

[1] Bellen, A. (1987): Parallelism across the steps for difference and differential equations, Lecture Notes in Mathematics 1386, Springer-Verlag, 22-35.

[2] Bellen, A., Jackiewicz, Z. and M. Zennaro (1990): Time-point relaxation Runge-Kutta methods for ordinary differential equations, J. Comp. Appl. Math. 45, 121-138.

[3] Bellen, A., Vermiglio, R. and M. Zennaro (1990): Parallel ODE solvers with step-size-control, J. Comp. Appl. Math. 31, 277-293.

[4] Burrage, K. (1993): Parallel methods for initial value problems, Appl. Numer. Math. 11, 5-26, 1993.

[5] Burrage, K. (1993): The search for the Holy Grail, or: Predictor-corrector methods for solving ODEIVPs, Appl. Numer. Math. 11, 125-141, 1993.

[6] Butcher, J.C. (1976): On the implementation of implicit Runge-Kutta methods, BIT 16, 237-240.

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

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

[9] Enright, W.H., Hull, T.E. & Lindberg, B. (1975): Comparing numerical methods for stiff systems of ODEs, BIT 15, 10-48.

[10] Gear, C.W. (1969): The automatic integration of stiff ordinary differential equations, Proc. IFIP Congress 1968, North Hollans, Amsterdam, 187-193.

[11] Gear, C.W. & Xu Xuhai (1993): Parallelism across time in ODEs, Appl. Numer. Math. 11, 45-68, 1993.

[12] Golub, G.H. & Van Loan, C.F. (1983): Matrix computations, North Oxford Academic, Oxford.

[13] Gottwald, B.A. (1977): MISS: A simple simulation system for biological and chemical processes (German), EDV in Medizin und Biologie 3, 85-90.

[14] Hairer, E. & Wanner, G. (1991): Solving ordinary differential equations, II. Stiff and differential-algebraic problems, Springer-Verlag, Berlin.

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

[16] Houwen, P.J. van der, Sommeijer, B.P. & W.A. van der Veen. (1995): Parallel iteration across the steps of high order Runge-Kutta methods for nonstiff initial value problems, JCAM.

[17] Houwen, P.J. van der, Sommeijer, B.P. & W.A. van der Veen. (1995): Parallelism across the steps in iterated Runge-Kutta methods for stiff initial value problems, Numerical Algorithms.

[18] Lioen,W.M. (1995): Parallel iteration of high-stage Radau IIA correctors, in preparation.

[19] Miranker, W.L. & Liniger, W. (1967): Parallel methods for the numerical integration of ordinary differential equations, Math. Comput. 21, 303-320.

[20] Orel, B. (1993): Parallel Runge-Kutta methods with real eigenvalues, Appl. Numer. Math. 11, 241-250, 1993.

[21] Verwer, J.G. (1994): Gauss-Seidel iteration for stiff ODEs from chemical kinetics, SIAM. J. Sci. Comput. 15, 1243-1250.

A. Appendix: Parameter matrices

For a number of RK methods, we have computed the matrices B = L + D according to the procedure outlined in Sections 3.2.1 and 3.2.2, together with the amplification matrices Z_0 and Z_∞.

A.1. PDIRK strategy: $Z(z) = z(I - zD)^{-1}(A - D)$, $Z_0 = A - D$, $Z_\infty = I - D^{-1}A$.

A.1.1. Radau IIA

$$s = 2: \quad B = \begin{pmatrix} 0.2584 & 0 \\ 0 & 0.6449 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} 0.1582 & -0.0833 \\ 0.7500 & -0.3949 \end{pmatrix}, \quad Z_\infty = \begin{pmatrix} -0.6124 & 0.3225 \\ -1.1629 & 0.6124 \end{pmatrix}.$$

$$s = 3: \quad B = \begin{pmatrix} 0.3204 & 0 & 0 \\ 0 & 0.1400 & 0 \\ 0 & 0 & 0.3717 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} -0.1236 & -0.0655 & 0.0238 \\ 0.3944 & 0.1521 & -0.0415 \\ 0.3764 & 0.5125 & -0.2606 \end{pmatrix},$$

$$Z_\infty = \begin{pmatrix} 0.3857 & 0.2045 & -0.0742 \\ -2.8179 & -1.0867 & 0.2968 \\ -1.0127 & -1.3789 & 0.7011 \end{pmatrix}.$$

$$s = 4: \quad B = \begin{pmatrix} 0.3205 & 0 & 0 & 0 \\ 0 & 0.0892 & 0 & 0 \\ 0 & 0 & 0.1817 & 0 \\ 0 & 0 & 0 & 0.2334 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} -0.2075 & -0.0403 & 0.0258 & -0.0099 \\ 0.2344 & 0.1177 & -0.0479 & 0.0160 \\ 0.2167 & 0.4061 & 0.0073 & -0.0242 \\ 0.2205 & 0.3882 & 0.3288 & -0.1709 \end{pmatrix},$$

$$Z_\infty = \begin{pmatrix} 0.6474 & 0.1258 & -0.0805 & 0.0309 \\ -2.6290 & -1.3206 & 0.5368 & -0.1800 \\ -1.1923 & -2.2346 & -0.0402 & 0.1331 \\ -0.9447 & -1.6635 & -1.4092 & 0.7322 \end{pmatrix}$$

A.1.2. Lobatto IIIA

$$s = 2: \quad B = \begin{pmatrix} 0.2113 & 0 \\ 0 & 0.3943 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} 0.1220 & -0.0417 \\ 0.6667 & -0.2277 \end{pmatrix}, \quad Z_\infty = \begin{pmatrix} -0.5774 & 0.1972 \\ -1.6906 & 0.5774 \end{pmatrix}.$$

$$s = 3: \quad B = \begin{pmatrix} 0.4802 & 0 & 0 \\ 0 & 0.1094 & 0 \\ 0 & 0 & 0.1604 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} -0.2905 & -0.0339 & 0.0103 \\ 0.4506 & 0.1175 & -0.0270 \\ 0.4167 & 0.4167 & -0.0770 \end{pmatrix},$$

$$Z_\infty = \begin{pmatrix} 0.6049 & 0.0706 & -0.0215 \\ -4.1179 & -1.0743 & 0.2465 \\ -2.5981 & -2.5981 & 0.4804 \end{pmatrix}.$$

A.1.3. Gauss

$$s = 2: \quad B = \begin{pmatrix} 0.1667 & 0 \\ 0 & 0.5000 \end{pmatrix}, \quad Z_0 = \begin{pmatrix} 0.0833 & -0.0387 \\ 0.5387 & -0.2500 \end{pmatrix}, \quad Z_\infty = \begin{pmatrix} -0.5000 & 0.2321 \\ -1.0774 & 0.5000 \end{pmatrix}.$$

A.2. PTIRK strategy: $Z(z) = z(I - zB)^{-1}(A - B)$, $Z_0 = A - B$, $Z_\infty = I - B^{-1}A$.

A.2.1. Radau IIA

s = 2: $B = \begin{pmatrix} 0.4167 & 0 \\ 0.7500 & 0.4000 \end{pmatrix}$, $Z_0 = \begin{pmatrix} 0 & -0.0833 \\ 0 & -0.1500 \end{pmatrix}$, $Z_\infty = \begin{pmatrix} 0 & 0.2000 \\ 0 & 0 \end{pmatrix}$.

s = 3: $B = \begin{pmatrix} 0.1968 & 0 & 0 \\ 0.3944 & 0.4234 & 0 \\ 0.3764 & 0.6378 & 0.2000 \end{pmatrix}$, $Z_0 = \begin{pmatrix} 0 & -0.0655 & 0.0238 \\ 0 & -0.1313 & -0.0415 \\ 0 & -0.1253 & -0.0889 \end{pmatrix}$,

$Z_\infty = \begin{pmatrix} 0 & 0.3330 & -0.1208 \\ 0 & 0 & 0.2106 \\ 0 & 0 & 0 \end{pmatrix}$.

s = 4: $B = \begin{pmatrix} 0.1130 & 0 & 0 & 0 \\ 0.2344 & 0.2905 & 0 & 0 \\ 0.2167 & 0.4834 & 0.3083 & 0 \\ 0.2205 & 0.4668 & 0.4414 & 0.1176 \end{pmatrix}$, $Z_0 = -\begin{pmatrix} 0 & 0.0403 & -0.0258 & 0.0099 \\ 0 & 0.0836 & 0.0479 & -0.0160 \\ 0 & 0.0773 & 0.1192 & 0.0242 \\ 0 & 0.0786 & 0.1126 & 0.0551 \end{pmatrix}$,

$Z_\infty = \begin{pmatrix} 0 & 0.3567 & -0.2283 & 0.0877 \\ 0 & 0 & 0.3490 & -0.1260 \\ 0 & 0 & 0 & 0.2144 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

A.2.2. Lobatto IIIA

s = 2: $B = \begin{pmatrix} 0.3333 & 0 \\ 0.6667 & 0.2500 \end{pmatrix}$, $Z_0 = \begin{pmatrix} 0 & -0.0417 \\ 0 & -0.0833 \end{pmatrix}$, $Z_\infty = \begin{pmatrix} 0 & 0.1250 \\ 0 & 0 \end{pmatrix}$.

s = 3: $B = \begin{pmatrix} 0.1897 & 0 & 0 \\ 0.4506 & 0.3075 & 0 \\ 0.4167 & 0.4911 & 0.1429 \end{pmatrix}$, $Z_0 = \begin{pmatrix} 0 & -0.0339 & 0.0103 \\ 0 & -0.0805 & -0.0270 \\ 0 & -0.0745 & -0.0595 \end{pmatrix}$,

$Z_\infty = \begin{pmatrix} 0 & 0.1787 & -0.0543 \\ 0 & 0 & 0.1673 \\ 0 & 0 & 0 \end{pmatrix}$.

A.2.3. Gauss

s = 2: $B = \begin{pmatrix} 0.2500 & 0 \\ 0.5387 & 0.3333 \end{pmatrix}$, $Z_0 = -\begin{pmatrix} 0 & 0.0387 \\ 0 & 0.0833 \end{pmatrix}$, $Z_\infty = \begin{pmatrix} 0 & 0.1547 \\ 0 & 0 \end{pmatrix}$.

B. Appendix: Costs of PDIRK and PTIRK

In this appendix we specify the costs of the implementations of PDIRK and PTIRK. For PTIRK there are three different implementations that were called previously PTIRK LJ version, PTIRK LF version and PTIRK transformed LJ version. We denote the average costs of one component of the right-hand-side function $\mathbf{f}$ and one entry of its Jacobian J by $C_{\mathbf{f}}$ and C_J, respectively. We assume that the Jacobian is full and that it is evaluated every ν time steps. As before, $\mathbf{W}$ is an sd-dimensional vector containing information computed in preceding integration steps and $\mathbf{R}(\mathbf{Y}^{(j)})$ and $\Delta\mathbf{Y}^{(j+1)}$ are defined by

$$\begin{aligned} \mathbf{R}(\mathbf{Y}^{(j)}) &= \mathbf{Y}^{(j)} - h(A \otimes I)\mathbf{F}(\mathbf{Y}^{(j)}) - \mathbf{W}, \\ \Delta\mathbf{Y}^{(j+1)} &= \mathbf{Y}^{(j+1)} - \mathbf{Y}^{(j)}. \end{aligned}$$

In the first column the computation that has to be performed is listed. The second column gives the number of floating point operations required for this computation if only one processor is available. The sequential costs of the compuation on s processors can be found in the third column.

PDIRK ((2.2) with $L = 0$):

$$(I - D \otimes hJ)\Delta\mathbf{Y}^{(j+1)} = -\mathbf{R}(\mathbf{Y}^{(j)}); \quad j = 0, \ldots, m-1$$

$$D = \mathrm{diag}(d_1, \ldots, d_s)$$

Computation	**Costs (flops)**	
	on 1 processor	on s processors
once per time step		
J	$d^2\ C_J$	$\frac{d^2}{s}\ C_J$
$I - hd_iJ$, $i = 1, \ldots, s$	sd^2	d^2
L_iU_i of $I - hd_iJ$, $i = 1, \ldots, s$	$\frac{2}{3}sd^3$	$\frac{2}{3}d^3$
once per iteration (for $j = 0, \ldots, m-1$)		
$\mathbf{F}(\mathbf{Y}^{(j)})$	$sd\ C_{\mathbf{f}}$	$d\ C_{\mathbf{f}}$
$\mathbf{R}(\mathbf{Y}^{(j)}) = \mathbf{Y}^{(j)} - h(A \otimes I)\mathbf{F}(\mathbf{Y}^{(j)}) - \mathbf{W}$	$2s^2d$	$2sd$
$\Delta\mathbf{Y}_i^{(j+1)} = -(L_iU_i)^{-1}\mathbf{R}(\mathbf{Y}_i^{(j)}),\ i = 1, \ldots, s$	$2sd^2$	$2d^2$
$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)}$	sd	d
Total per time step	$\frac{1}{\nu}sd^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$	$\frac{1}{\nu}d^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$
after m iterations	$+msd\ C_{\mathbf{f}}$	$+md\ C_{\mathbf{f}}$
	$+msd(2d + 2s)$	$+md(2d + 2s)$

PTIRK LJ version ((2.2) with $L \neq 0$):

$(I - D \otimes hJ)\Delta\mathbf{Y}^{(j+1)} = (L \otimes hJ)\Delta\mathbf{Y}^{(j+1)} - \mathbf{R}(\mathbf{Y}^{(j)}); \quad j = 0, \ldots, m-1$

$B = D + L; \quad D = \mathrm{diag}(d_1, \ldots, d_s); \quad L = (l_{ij}); \quad L$ strictly lower triangular

Computation	**Costs (flops)**	
	on 1 processor	on s processors
once per time step		
J	$d^2\ C_J$	$\frac{d^2}{s}\ C_J$
$I - hd_iJ$, $i = 1, \ldots, s$	sd^2	d^2
L_iU_i of $I - hd_iJ$, $i = 1, \ldots, s$	$\frac{2}{3}sd^3$	$\frac{2}{3}d^3$
once per iteration (for $j = 0, \ldots, m-1$)		
$\mathbf{F}(\mathbf{Y}^{(j)})$	$sd\ C_{\mathbf{f}}$	$d\ C_{\mathbf{f}}$
$\mathbf{R}(\mathbf{Y}^{(j)}) = \mathbf{Y}^{(j)} - h(A \otimes I)\mathbf{F}(\mathbf{Y}^{(j)}) - \mathbf{W}$	$2s^2d$	$2sd$
$\widetilde{\mathbf{R}}_i^{(j)} = \mathbf{R}_i^{(j)} - \sum_{k=1}^{i-1} l_{ik}hJ\Delta\mathbf{Y}_k^{(j+1)}$,		
$i = 1, \ldots, s$	$2sd^2 + s^2d$	$2sd^2 + s^2d$
$\Delta\mathbf{Y}_i^{(j+1)} = -(L_iU_i)^{-1}\widetilde{\mathbf{R}}(\mathbf{Y}_i^{(j)})$, $i = 1, \ldots, s$	$2sd^2$	$2sd^2$
$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)}$	sd	d
Total per time step	$\frac{1}{\nu}sd^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$	$\frac{1}{\nu}d^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$
after m iterations	$+msd\ C_{\mathbf{f}}$	$+md\ C_{\mathbf{f}}$
	$+msd(4d + 3s)$	$+md(4sd + s^2)$

PTIRK LF version ((2.3) with $L \neq 0$):

$(I - D \otimes hJ)\Delta\mathbf{Y}^{(j+1)} = L \otimes h(\mathbf{F}(\mathbf{Y}^{(j+1)}) - \mathbf{F}(\mathbf{Y}^{(j)})) - \mathbf{R}(\mathbf{Y}^{(j)}); \quad j = 0, \ldots, m-1$

$B = D + L; \quad D = \mathrm{diag}(d_1, \ldots, d_s); \quad L = (l_{ij}); \quad L$ strictly lower triangular

Computation	**Costs (flops)**	
	on 1 processor	on s processors
once per time step		
J	$d^2\ C_J$	$\frac{d^2}{s}\ C_J$
$I - hd_iJ$, $i = 1, \ldots, s$	sd^2	d^2
L_iU_i of $I - hd_iJ$, $i = 1, \ldots, s$	$\frac{2}{3}sd^3$	$\frac{2}{3}d^3$
once per iteration (for $j = 0, \ldots, m-1$)		
$\mathbf{R}(\mathbf{Y}^{(j)}) = \mathbf{Y}^{(j)} - h(A \otimes I)\mathbf{F}(\mathbf{Y}^{(j)}) - \mathbf{W}$	$2s^2d$	$2sd$
$\widetilde{\mathbf{R}}_i^{(j)} = \mathbf{R}_i^{(j)} - \sum_{k=1}^{i-1} l_{ik}h(\mathbf{F}(\mathbf{Y}_k^{(j+1)}) - \mathbf{F}(\mathbf{Y}^{(j)}))$, $i = 1, \ldots, s$	$sd + s^2d$	$sd + s^2d$
$\Delta\mathbf{Y}_i^{(j+1)} = -(L_iU_i)^{-1}\widetilde{\mathbf{R}}(\mathbf{Y}_i^{(j)})$, $i = 1, \ldots, s$	$2sd^2$	$2sd^2$
$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)}$	sd	d
$\mathbf{F}(\mathbf{Y}^{(j+1)})$	$sd\ C_{\mathbf{f}}$	$sd\ C_{\mathbf{f}}$
Total per time step	$\frac{1}{\nu}sd^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$	$\frac{1}{\nu}d^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$
after m iterations	$+msd\ C_{\mathbf{f}}$	$+msd\ C_{\mathbf{f}}$
	$+msd(2d + 3s)$	$+md(2sd + s^2)$

PTIRK transformed LJ version (2.4):

$$\left.\begin{aligned} &(I - D \otimes hJ)\Delta\mathbf{X}^{(j+1)} = -(Q^{-1} \otimes I)\mathbf{R}(\mathbf{Y}^{(j)}) \\ &\Delta\mathbf{Y}^{(j+1)} = (Q \otimes I)\Delta\mathbf{X}^{(j+1)} \end{aligned}\right\} \quad j = 0, \ldots, m-1$$

$BQ = QD; \quad D = \mathrm{diag}(d_1, \ldots, d_s); \quad B$ and Q lower triangular

Computation	**Costs (flops)**	
	on 1 processor	on s processors
once per time step		
J	$d^2\ C_J$	$\frac{d^2}{s}\ C_J$
$I - hd_iJ$, $i = 1, \ldots, s$	sd^2	d^2
L_iU_i of $I - hd_iJ$, $i = 1, \ldots, s$	$\frac{2}{3}sd^3$	$\frac{2}{3}d^3$
once per iteration (for $j = 0, \ldots, m-1$)		
$\mathbf{F}(\mathbf{Y}^{(j)})$	$sd\ C_{\mathbf{f}}$	$d\ C_{\mathbf{f}}$
$\mathbf{R}(\mathbf{Y}^{(j)}) = \mathbf{Y}^{(j)} - h(A \otimes I)\mathbf{F}(\mathbf{Y}^{(j)}) - \mathbf{W}$	$2s^2d$	$2sd$
$\widetilde{\mathbf{R}}(\mathbf{Y}^{(j)}) = (Q^{-1} \otimes I)\mathbf{R}(\mathbf{Y}^{(j)})$	s^2d	sd
$\Delta\mathbf{X}_i^{(j+1)} = -(L_iU_i)^{-1}\widetilde{\mathbf{R}}(\mathbf{Y}_i^{(j)}),\ i = 1, \ldots, s$	$2sd^2$	$2d^2$
$\Delta\mathbf{Y}^{(j+1)} = (Q \otimes I)\Delta\mathbf{X}^{(j+1)}$	s^2d	sd
$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} + \Delta\mathbf{Y}^{(j+1)}$	sd	d
Total per time step	$\frac{1}{\nu}sd^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$	$\frac{1}{\nu}d^2(\frac{1}{s}\ C_J + \frac{2}{3}d + 1)$
after m iterations	$+msd\ C_{\mathbf{f}}$	$+md\ C_{\mathbf{f}}$
	$+msd(2d + 4s)$	$+md(2d + 4s)$