MATHEMATICS OF COMPUTATION
VOLUME 58, NUMBER 197
JANUARY 1992, PAGES 135–159

EMBEDDED DIAGONALLY IMPLICIT RUNGE-KUTTA ALGORITHMS ON PARALLEL COMPUTERS

P. J. VAN DER HOUWEN, B. P. SOMMEIJER, AND W. COUZY

ABSTRACT. This paper investigates diagonally implicit Runge-Kutta methods in which the implicit relations can be solved in parallel and are singly diagonal-implicit on each processor. The algorithms are based on diagonally implicit iteration of fully implicit Runge-Kutta methods of high order. The iteration scheme is chosen in such a way that the resulting algorithm is $A(\alpha)$-stable or $L(\alpha)$-stable with α equal or very close to $\pi/2$. In this way, highly stable, singly diagonal-implicit Runge-Kutta methods of orders up to 10 can be constructed. Because of the iterative nature of the methods, embedded formulas of lower orders are automatically available, allowing a strategy for step and order variation.

1. INTRODUCTION

In Nørsett and Simonsen [21], Jackson and Nørsett [16], and Iserles and Nørsett [15], it was observed that on parallel computers, predictor-corrector methods (PC methods) based on implicit Runge-Kutta (RK) correctors are particularly attractive for solving initial value problems for the system of ordinary differential equations (ODE's)

$$\frac{dy(t)}{dt} = f(y(t)). \tag{1.1}$$

On sequential computers, implicit RK methods are seldom used as corrector equations, because of the large number of implicit relations to be solved when using these correctors. However, matters are different when parallel computers are used, since PC methods, being a form of functional iteration, possess a high degree of parallelism. First results based on the PC approach were reported by Lie [18], who uses a fourth-order, two-stage Gauss-Legendre corrector and a third-order Hermite extrapolation predictor. In [12], these "parallel, iterated" RK methods (which we shall briefly call *PIRK methods*) have been investigated for a variety of predictor methods and it was concluded that, from an implementational point of view, one-step predictors are preferable. Related PC methods were studied by Tam in his thesis [24]. In particular, families of methods were constructed with elliptically shaped stability regions. An analysis of the error behavior of a very general class of PC methods, including all methods indicated above, was given by Burrage [2].

Received June 26, 1989; revised May 17, 1990.
1991 *Mathematics Subject Classification.* Primary 65M12, 65M20.
Key words and phrases. Runge-Kutta methods, parallelism.

An attractive feature of PIRK methods is the availability of embedded formulas of lower orders allowing a strategy for step and order variation without additional costs. On the other hand, owing to their explicit character, PIRK methods have rather limited regions of stability and are therefore only suitable for integrating *nonstiff* systems.

In this paper, we shall be interested in integrating *stiff* systems, and we will investigate the possibility of constructing methods that are more stable than PIRK methods by *diagonally implicit iteration* of fully implicit RK methods. After a fixed number of iterations, such methods belong to the class of DIRK methods, and are therefore essentially different from the explicit PIRK methods studied in the aforementioned papers. DIRK methods resulting from diagonally implicit iteration have the property that *effectively* they are singly diagonal-implicit RK (SDIRK) methods when run on parallel computers. Furthermore, like the PIRK methods, they possess embedded formulas of lower order, which make them an ideal starting point for developing variable order/variable step codes. We shall call the "Parallel Diagonal-implicitly Iterated" RK methods *PDIRK methods.*

In the literature, various (S)DIRK methods were published for the integration of stiff systems of ODE's. The most recent contributions are the parallel DIRK methods of Iserles and Nørsett [15], which are, like PDIRK methods, effectively of SDIRK-type on multi-processor computers (these methods are the first and, as far as we know, the only parallel DIRK methods published in the literature). However, the order of most DIRK methods is limited to $p = 4$ (the only DIRK methods exceeding this order are those of Cooper and Sayfy, see this Journal, Vol. 33, 1979, pp. 541–556). By diagonal iteration of implicit RK methods it is possible to construct highly stable PDIRK methods of orders up to 10.

Table 1.1 presents the characteristics of a number of SDIRK methods from the literature, together with the most stable PDIRK methods of order $p > 4$ derived in the present paper. In this table, DIRK II denotes the Type II methods of Iserles and Nørsett [15], p_{emb} indicates that embedded methods of orders $\leq p_{\text{emb}}$ are available, and s denotes the number of stages of the underlying corrector in the PDIRK methods (by choosing Gauss-Legendre or Radau IIA correctors, we may set $s = [(p+1)/2]$, where $[\cdot]$ denotes the integer-part function). Furthermore, the number of sequential stages is defined as the number of implicit systems to be solved on each processor in each step. Finally, we introduce the concept of L^2-stability, which means that the method possesses an A-acceptable stability function for which the degree of the numerator is two less than the degree of the denominator.

This table shows that the PDIRK methods constructed in this paper have the advantages of high order, high stability, and embedded formulas, but the disadvantage of quite a large number of sequential stages per step. For example, in spite of its inherent parallelism, the number of sequential stages per step of an L^2-stable, eighth-order PDIRK method is three times as large as that of the A-stable, fourth-order SDIRK method of Crouzeix and Alexander, and nine times as large as that of the BDF methods. However, because of the iterative nature of PDIRK methods, the "later" stages are relatively cheap since there are accurate initial iterates available for solving the associated implicit relations. This feature, and in particular their high order and unconditional stability, make PDIRK methods a promising starting point to base a code on. This is confirmed

TABLE 1.1
(S)DIRK and PDIRK methods

Method	Order	Stages	Seq. stages	Processors	Stability	p_{emb}	Reference
SDIRK	$p = 3$	$p - 1$	$p - 1$	1	A-stable	1	[19]
SDIRK	$p = 3$	$p - 1$	$p - 1$	1	Strongly A-stable	1	[6]
SDIRK	$p = 4$	$p - 1$	$p - 1$	1	A-stable	1	[6], [1]
SDIRK	$p = 3$	p	p	1	S-stable	$p - 1$	[4], [5]
SDIRK	$p = 3$	$p + 1$	$p + 1$	1	L-stable	$p - 1$	[22]
SDIRK	$p = 4$	$p + 1$	$p + 1$	1	S-stable	$p - 1$	[4], [5]
DIRK II	$p = 4$	p	$p - 2$	2	L-stable	$p - 1$	[15]
PDIRK	$p = 5$	$3(p - 1)$	$p - 1$	3	Strongly A-stable	$p - 1$	§3.2
PDIRK	$p = 6$	$3(p - 1)$	$p - 1$	3	Strongly $A(\alpha)$-stable	$p - 1$	$\alpha \approx 89.97°$: §3.2
PDIRK	$p = 7$	$4(p - 1)$	$p - 1$	4	$A(\alpha)$-stable	$p - 1$	$\alpha \approx 89.95°$: §3.2
PDIRK	$p \leq 6$	sp	p	s	L-stable	$p - 1$	§3.1
PDIRK	$p = 8$	sp	p	s	L-stable	$p - 1$	§3.1
PDIRK	$p \leq 8$	$s(p + 1)$	$p + 1$	s	L^2-stable	$p - 1$	§3.1
PDIRK	$p = 10$	$s(p + 1)$	$p + 1$	s	L^2-stable	$p - 1$	§3.1

by a few preliminary experiments reported in §4, where we show by means of two "difficult" test problems taken from the literature that a provisional implementation of an L^2-stable, seventh-order, four-processor PDIRK method is already far superior to the SDIRK code SIMPLE of Nørsett and Thomsen [22] and at least competitive with the BDF code LSODE of Hindmarsh [11]. The development of a more sophisticated code based on PDIRK-type methods and much more extensive comparisons with existing sequential codes on a significant class of stiff problems will be the subject of our future research and should provide more reliable data on the efficiency of PDIRK-based codes.

2. PDIRK METHODS

For notational convenience, we shall assume in the following that equation (1.1) is a scalar equation. However, all considerations below are straightforwardly extended to systems of ODE's, and therefore also to nonautonomous equations. Our starting point is the s-stage, implicit, one-step RK method

$$y_{n+1} = y_n + h\mathbf{b}^T f(\mathbf{Y}), \tag{2.1a}$$

where $\mathbf{Y}$ is implicitly defined by the set of algebraic equations

$$\mathbf{Y} := y_n\mathbf{e} + hAf(\mathbf{Y}). \tag{2.1b}$$

Here, h is the integration step, $\mathbf{e}$ is a column vector of dimension s with unit entries, $\mathbf{b}$ is an s-dimensional vector, and A is an s-by-s matrix. Furthermore, we use the convention that for any given vector $\mathbf{v} = (v_j)$, $f(\mathbf{v})$ denotes the vector with entries $f(v_j)$.

By iterating, say m times, the equation for $\mathbf{Y}$ by diagonally implicit iteration, we obtain the method

$$\begin{aligned} \mathbf{Y}^{(j)} &= y_n\mathbf{e} + h[A - D]f(\mathbf{Y}^{(j-1)}) + hDf(\mathbf{Y}^{(j)}), \\ y^{(j)} &= y_n + h\mathbf{b}^{\mathrm{T}}f(\mathbf{Y}^{(j)}), \qquad j = 1, 2, \ldots, m, \end{aligned} \tag{2.2}$$

where D is a diagonal matrix with arbitrary, nonnegative diagonal elements and $\mathbf{Y}^{(0)}$ denotes an initial approximation to the vector $\mathbf{Y}$. Notice that after each iteration the current approximation $y^{(j)}$ to y_{n+1} can be computed. As we shall see in §2.1, the order of these approximations increases by 1 in each iteration. Therefore, the mth iterate will be used to continue the integration process and the preceding iterates can be used for error control.

Since the matrix D is of diagonal form, the s components of each vector $\mathbf{Y}^{(j)}$ can be computed in parallel, provided that s processors are available. Thus, effectively, we obtain a method which requires per integration step the computational time needed for computing one component of the initial approximation $\mathbf{Y}^{(0)}$ and the successive solution of m equations. In the following, we always assume that we have s processors at our disposal, and we shall speak about computational effort per step when we mean the computational time required per step if s processors are available. We shall call the method providing $\mathbf{Y}^{(0)}$ the *predictor method* and (2.1) the *corrector method*.

There are several possibilities for choosing the matrix D. The simplest choice sets $D = 0$ to obtain an explicit iteration method (fixed point or functional iteration). This approach was followed in, e.g., Nørsett and Simonsen [21], Lie [18], and van der Houwen and Sommeijer [12]. These papers deal with the iteration of implicit methods for solving *nonstiff* ODE's. As stated in the introduction, we are aiming at *stiff* ODE's, which requires the use of matrices $D \neq 0$. One possibility of exploiting nonzero matrices D is improving the rate of convergence of the iteration process. For example, by identifying the diagonal elements of D with those of A, we obtain the nonlinear Jacobi iteration method. Alternatively, one may choose D such that the stability region of the iterated method rapidly converges to that of the corrector (cf. [13]). In this paper, however, we choose D such that we have for a prescribed number of iterations favorable stability characteristics, such as A-stability or L-stability (as far as we know, this approach has not yet been investigated in the literature). We restrict our considerations to the case where the predictor method is itself an RK-type method. Hence, by performing m iterations with (2.2) and by accepting $y^{(m)}$ as the final approximation to y_{n+1}, we obtain an RK method with a fixed number of stages. Furthermore, we assume that the predictor is explicit or at most diagonally implicit. Then, the resulting parallel RK method belongs to the class of DIRK methods (Diagonally Implicit RK methods), and will be briefly called the *PDIRK method*.

2.1. **Order of PDIRK methods.** Assuming that the iteration process (2.2) converges as $m \to \infty$, the values $y^{(j)}$ approximate the solution of the corrector method (2.1), i.e., $y^{(\infty)} = y_{n+1}$. The approximation $y^{(j)}$ differs from $y^{(\infty)}$ by the amount

$$y^{(j)} - y^{(\infty)} = y^{(j)} - y_{n+1} = h\mathbf{b}^{\mathrm{T}}[f(\mathbf{Y}^{(j)}) - f(\mathbf{Y})].$$

If the right-hand side function is sufficiently smooth, then the iteration error $\mathbf{Y}^{(j)} - \mathbf{Y}$ satisfies the approximate recursion

$$\begin{aligned}\mathbf{Y}^{(j)} - \mathbf{Y} &\approx h\left[I - h\frac{\partial f}{\partial y}D\right]^{-1}\frac{\partial f}{\partial y}[A - D][\mathbf{Y}^{(j-1)} - \mathbf{Y}] \\ &= h^j\left(\left[I - h\frac{\partial f}{\partial y}D\right]^{-1}\frac{\partial f}{\partial y}[A - D]\right)^j[\mathbf{Y}^{(0)} - \mathbf{Y}],\end{aligned}$$

so that

$$y^{(m)} - y_{n+1} \approx h^{m+1}\frac{\partial f}{\partial y}\mathbf{b}^{\mathrm{T}}\left(\left[I - h\frac{\partial f}{\partial y}D\right]^{-1}\frac{\partial f}{\partial y}[A - D]\right)^m[\mathbf{Y}^{(0)} - \mathbf{Y}]. \tag{2.3}$$

Let the predictor be of order q, i.e.,

$$\mathbf{Y}^{(0)} - \mathbf{Y} = O(h^q) \Rightarrow y^{(0)} - y_{n+1} = O(h^{q+1}); \tag{2.4}$$

then

$$y^{(m)} - y_{n+1} = O(h^{q+m+1}),$$

so that $y^{(m)}$ has (global) order $q + m$.

In this paper, we shall study PDIRK methods with predictors of the form

$$\mathbf{Y}^{(0)} := y_n\mathbf{e} + hEf(y_n\mathbf{e}) + hBf(\mathbf{Y}^{(0)}). \tag{2.5}$$

Because this predictor is *implicit*, we will choose the matrix B of diagonal form in order to exploit parallelism. Since

$$\begin{aligned}\mathbf{Y}^{(0)} - \mathbf{Y} = y_n\mathbf{e} &+ hEf(y_n\mathbf{e}) + hBf(y_n\mathbf{e} + hEf(y_n\mathbf{e}) + hBf(y_n\mathbf{e})) \\ &- y_n\mathbf{e} - hAf(y_n\mathbf{e} + hAf(y_n\mathbf{e})) + O(h^3),\end{aligned}$$

it is easily verified that the predictor (2.5) is always first-order accurate; it becomes of order two if $(E + B - A)\mathbf{e}$ vanishes, and of order three if, in addition, $(BA - A^2)\mathbf{e}$ vanishes.

By defining y_{n+1} according to

$$y_{n+1} := y^{(m)} = y_n + h\mathbf{b}^{\mathrm{T}}f(\mathbf{Y}^{(m)}), \tag{2.6}$$

the PDIRK method is completely determined. For this method, we summarize the above order considerations in the following theorem.

Theorem 2.1. *Let the corrector be of order* p^*; *then the approximation* y_{n+1} *generated by the PDIRK method* {(2.5), (2.2), (2.6)} *has order* $\min\{p^*, m+1\}$ *for all matrices* B *and* E, *order* $\min\{p^*, m+2\}$ *if* $(E+B)\mathbf{e} = A\mathbf{e}$, *and order* $\min\{p^*, m+3\}$ *if, in addition,* $BA\mathbf{e} = A^2\mathbf{e}$.

We remark that correctors of any order are explicitly available. Correctors of any even order p^* are provided by the $(p^*/2)$-stage Gauss-Legendre methods,

and correctors of any odd order p^* are provided by the $((p^*+1)/2)$-stage Radau methods.

2.2. **Stiffly accurate PDIRK methods.** As was discussed by Alexander [1], when integrating stiff equations, it may be advantageous to use RK methods $\{A, \mathbf{b}\}$ in which $\mathbf{b}$ equals the last row of A, i.e., $\mathbf{b}^T = \mathbf{e}_s^T A$, where s is the number of stages of the RK method. Such RK methods are termed *stiffly accurate.* Therefore, it is of interest to look for PDIRK methods possessing the property of stiff accuracy. Formally, we can associate with any PDIRK method a new PDIRK method possessing the property of stiff accuracy, simply by replacing (2.6) with

$$y_{n+1} = \mathbf{e}_s^T \mathbf{Y}^{(m)}. \tag{2.7}$$

Of course, this only yields a feasible method if the last component of the vector $\mathbf{Y}^{(m)}$ provides an approximation to y_{n+1}. For example, this is true if the corrector itself is stiffly accurate, i.e., $\mathbf{b}^T = \mathbf{e}_s^T A$. We shall call the two versions corresponding to (2.6) and (2.7) PDIRK methods of Type I and II, and denote them by PDIRK$^{\mathrm{I}}$ and PDIRK$^{\mathrm{II}}$, respectively. Thus,

Type I: PDIRK method {(2.5), (2.2), (2.6)},
Type II: PDIRK method {(2.5), (2.2), (2.7)}.

The following theorem is the analogue of Theorem 2.1.

Theorem 2.2. *Let the corrector be stiffly accurate* ($\mathbf{b}^T = \mathbf{e}_s^T A$) *and be of order* p^*; *then the approximation* y_{n+1} *generated by the PDIRK*$^{\mathrm{II}}$ *method is also stiffly accurate and has order* $\min\{p^*, m\}$ *for all matrices* B *and* E, *order* $\min\{p^*, m+1\}$ *if* $(E+B)\mathbf{e} = A\mathbf{e}$, *and order* $\min\{p^*, m+2\}$ *if, in addition,* $BA\mathbf{e} = A^2\mathbf{e}$.

2.3. **Various types of PDIRK methods and their Butcher arrays.** Given the generating RK method (corrector) $\{A, \mathbf{b}\}$ defined by (2.1), we shall investigate three special families of PDIRK methods, either of Type I or of Type II, which differ from each other by the way in which the predictor is defined, i.e., the matrices B and E are chosen. Let O denote the s-by-s matrix with zero entries; then we distinguish:

Type A: Last-step-value predictor $(E = B = O)$ $\mathbf{Y}^{(0)} := y_n\mathbf{e}$,

Type B: Backward Euler predictor $(E = O, B = D)$ $\mathbf{Y}^{(0)} := y_n\mathbf{e} + hDf(\mathbf{Y}^{(0)})$,

Type C: Theta method predictor $(B = D)$ $\mathbf{Y}^{(0)} := y_n\mathbf{e} + hEf(y_n\mathbf{e}) + hDf(\mathbf{Y}^{(0)})$.

Notice that the matrix B either vanishes or is chosen equal to D. Although, in general, B and D may be different (diagonal) matrices, the particular choice $B = D$ has advantages with respect to the implementation of the method. Typically for stiff equations, the implicit relations in which the matrix $D = \mathrm{diag}(d_1, d_2, \ldots, d_s)$ is involved will be solved by some form of Newton iteration, which requires (in the case of systems of ODE's) the LU-decomposition of the matrices $I - d_i h \partial f/\partial y$. Clearly, if $B = D$ then these decompositions can also be used in solving the predictor (see also the discussion below). In the remainder of this paper, the analysis is performed in terms of a general matrix B, and concrete results are only specified for $B = O$ or $B = D$.

For future reference, we specify the various PDIRK$^{\mathrm{I}}$ families of methods in

terms of their Butcher arrays and give the corresponding orders of accuracy p^{I}:

Type IA:

1. $D \neq O$: $p^{\mathrm{I}} = \min\{p^*, m+1\}$

$j=0$	O						
$j=1$	$A-D$	D					
$j=2$	O	$A-D$	D				
$j=3$	O	O	$A-D$	D			
$\vdots$	$\vdots$		$\ddots$	$\ddots$	$\ddots$		
$j=m$	O	$\cdots$	$\cdots$	$\cdots$	O	$A-D$	D
	$\mathbf{0}^{\mathrm{T}}$	$\cdots$	$\cdots$	$\cdots$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{b}^{\mathrm{T}}$

Type IB:

1. $D \neq O$: $p^{\mathrm{I}} = \min\{p^*, m+1\}$
2. $D := \mathrm{diag}(A\mathbf{e})$: $p^{\mathrm{I}} = \min\{p^*, m+2\}$

$j=0$	D						
$j=1$	$A-D$	D					
$j=2$	O	$A-D$	D				
$j=3$	O	O	$A-D$	D			
$\vdots$	$\vdots$		$\ddots$	$\ddots$	$\ddots$		
$j=m$	O	$\cdots$	$\cdots$	$\cdots$	O	$A-D$	D
	$\mathbf{0}^{\mathrm{T}}$	$\cdots$	$\cdots$	$\cdots$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{b}^{\mathrm{T}}$

Type IC:

1. $D \neq O$, $E \neq O$: $p^{\mathrm{I}} = \min\{p^*, m+1\}$
2. $D := \mathrm{diag}(A\mathbf{e} - E\mathbf{e})$, $E \neq O$: $p^{\mathrm{I}} = \min\{p^*, m+2\}$
3. $D := \mathrm{diag}(A\mathbf{e} - E\mathbf{e})$, $DA\mathbf{e} = A^2\mathbf{e}$: $p^{\mathrm{I}} = \min\{p^*, m+3\}$

	O						
$j=0$	E	D					
$j=1$	O	$A-D$	D				
$j=2$	O	O	$A-D$	D			
$\vdots$	$\vdots$		$\ddots$	$\ddots$	$\ddots$		
$j=m$	O	$\cdots$	$\cdots$	$\cdots$	O	$A-D$	D
	$\mathbf{0}^{\mathrm{T}}$	$\cdots$	$\cdots$	$\cdots$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{0}^{\mathrm{T}}$	$\mathbf{b}^{\mathrm{T}}$

In these arrays, $\mathbf{0}$ denotes the s-dimensional nullvector. Type II versions are obtained by defining y_{n+1} by means of (2.7) instead of by (2.6), and, if the weights of the corrector satisfy $\mathbf{b}^{\mathrm{T}} = \mathbf{e}_s^{\mathrm{T}} A$, then by virtue of Theorem 2.2, we may replace p^{I} by p^{II} and m by $m-1$. Notice that the $\mathbf{b}$-vector is not actually needed if the algorithm is based on Type II methods. Furthermore, we remark that methods of Type B.2 are completely determined by the generating corrector, and that those of Type C.3 prescribe the matrix D and the row sums of the matrix E.

As already observed, PDIRK methods all belong to the class of DIRK methods (since the name DIRK is not consistently used in the literature, we remark that we shall call an RK method of DIRK type if the strictly upper

triangular part of its Butcher tableau vanishes). Moreover, the ith processor $(i = 1, 2, \ldots, s)$ is faced with solving a sequence of implicit relations in each of which the decomposition of the matrix $I - d_i h \partial f/\partial y$ is required (in case of systems of ODE's). Since this decomposition can be used in all m iterations in (2.2), we shall say that PDIRK methods are *singly* diagonally implicit RK methods (SDIRK methods). Here we remark that this terminology is often reserved for methods in which *all* stages are implicit with the same diagonal entry in their Butcher array. However, the zero diagonal entries in PDIRK methods of Types A and C (originating from $B = O$) do not exclude these methods from the class of SDIRK methods, since these zeros mean that $f(y_n)$ has to be evaluated prior to the iteration process. Because the bulk of the computational effort per step consists in solving the implicit relations, the costs of this explicit stage are relatively negligible.

Therefore, taking parallelism into account, we shall say that PDIRK methods require k *sequential stages* if each processor has to solve k implicit relations per step. Thus, Type A methods require m sequential stages, whereas for Type B and Type C methods this number is given by $m + 1$.

Finally, we observe that if the diagonal matrix D has *equal* diagonal entries, then all processors need the same LU-decomposed matrix in their solution processes. In such cases, this decomposition, as well as the evaluation of the Jacobian matrix $\partial f/\partial y$, may be performed by an additional processor, providing a "fresh" decomposition for *all* processors as soon as it is available.

3. Stability

Applying the PDIRK method to the test equation

$$y'(t) = \lambda y(t) \tag{3.1}$$

yields a relation of the form $y_{n+1} = R_m(z) y_n$, where $z := \lambda h$ and $R_m(z)$ is a rational function, the so-called *stability function*. The stability functions corresponding to PDIRK$^{\mathrm{I}}$ and PDIRK$^{\mathrm{II}}$ methods will be denoted by $R_m^{\mathrm{I}}(z)$ and $R_m^{\mathrm{II}}(z)$, respectively. They can be directly derived from the Butcher arrays by using the familiar "determinant formula" (cf., e.g., [7, p. 72]). However, the dimension of these arrays is usually so high that the evaluation of the determinants is rather tedious, even for small values of the number of iterations m. Therefore, we shall derive alternative formulas.

From (2.6) and (2.7) we see that the stability functions are respectively determined by

$$y_{n+1} = y_n + z\mathbf{b}^{\mathrm{T}}\mathbf{Y}^{(m)} = R_m^{\mathrm{I}}(z) y_n \quad \text{and} \quad y_{n+1} = \mathbf{e}_s^{\mathrm{T}}\mathbf{Y}^{(m)} = R_m^{\mathrm{II}}(z) y_n. \tag{3.2}$$

In order to derive an expression for $\mathbf{Y}^{(m)}$, we write

$$\mathbf{Y}^{(j)} = [I - zD]^{-1} Q_j y_n \mathbf{e},$$

where the matrix Q_j follows from

$$\begin{aligned} \mathbf{Y}^{(j)} &= [I - zD]^{-1}[y_n \mathbf{e} + z(A - D)\mathbf{Y}^{(j-1)}] \\ &= [I - zD]^{-1}[y_n \mathbf{e} + z(A - D)[I - zD]^{-1} Q_{j-1} y_n \mathbf{e}]. \end{aligned}$$

Introducing the matrix function $Z = Z(z) := z(A - D)(I - zD)^{-1}$, we find that Q_j satisfies the recursion

$$Q_0 = [I - zD][I - zB]^{-1}[I + zE], \qquad Q_j = I + ZQ_{j-1}.$$

Hence, the stability functions are given by

$$R_m^{\mathrm{I}}(z) = 1 + z\mathbf{b}^{\mathrm{T}}[I - zD]^{-1}Q_m(z)\mathbf{e}, \quad R_m^{\mathrm{II}}(z) = \mathbf{e}_s^{\mathrm{T}}[I - zD]^{-1}Q_m(z)\mathbf{e},$$

$$Q_m = Q_m(z) := I + Z + Z^2 + \cdots + Z^{m-1} + Z^m[I - zD][I - zB]^{-1}[I + zE]. \tag{3.3}$$

We shall separately consider the case where the diagonal matrices B and D have *constant* diagonal elements, and the case where the matrices B and D are *arbitrary* diagonal matrices.

3.1. **PDIRK methods with constant diagonal elements.** First, we consider the effect of setting $D = dI$ on the attainable order of those PDIRK methods which already impose conditions on the matrix D. Assuming that the generating corrector always satisfies the condition $A\mathbf{e} = \mathbf{c}$, we find, according to the specification of PDIRK methods in §2.3, that

Type B.2: $D = \mathrm{diag}(A\mathbf{e}) \Rightarrow d\mathbf{e} = \mathbf{c}$,

Type C.3: $DA\mathbf{e} = A^2\mathbf{e} \Rightarrow d\mathbf{c} = A\mathbf{c}$.

By observing that third-order correctors require $\mathbf{b}^{\mathrm{T}}\mathbf{e} = 1$, $\mathbf{b}^{\mathrm{T}}\mathbf{c} = \frac{1}{2}$, $\mathbf{b}^{\mathrm{T}}A\mathbf{c} = \frac{1}{6}$, and $\mathbf{b}^{\mathrm{T}}\mathbf{c}^2 = \frac{1}{3}$, we see that PDIRK methods of Type B.2 cannot satisfy these conditions, so that their order is limited to $p^* = 2$, which is obtained for $d = \frac{1}{2}$. A necessary condition for Type C.3 methods to satisfy these third-order conditions requires $d = \frac{1}{3}$. However, the fourth-order condition $\mathbf{b}^{\mathrm{T}}A^2\mathbf{c} = \frac{1}{24}$ cannot be satisfied, so that the order of Type C.3 methods is limited to $p^* = 3$. Obviously, we are not interested in such low-order methods. Furthermore, as will be shown below, we shall exclude methods of Type C.1, because the number of sequential stages is not optimal with respect to the order p. Thus, in this section we shall concentrate on PDIRK methods of Type A.1, Type B.1, and Type C.2.

Next, we return to the stability functions (3.3). For $B = bI$ and $D = dI$, the matrix $Q_m(z)$ can be written as

$$Q_m(z) = \frac{N_m(z)}{(1 - bz)(1 - dz)^{m-1}},$$

where $N_m(z)$ is a polynomial in z with matrix-valued coefficients; (3.3) becomes

$$R_m^{\mathrm{I}}(z) = 1 + \frac{\mathbf{b}^{\mathrm{T}}zN_m(z)\mathbf{e}}{(1 - bz)(1 - dz)^m}, \quad R_m^{\mathrm{II}}(z) = \frac{\mathbf{e}_s^{\mathrm{T}}N_m(z)\mathbf{e}}{(1 - bz)(1 - dz)^m}. \tag{3.4}$$

This representation shows that both stability functions are of the form

$$R(z) := (1 - dz)^{-q}P(dz), \qquad P(dz) := \sum_{j=0}^{r} c_j(dz)^j, \tag{3.5a}$$

where the coefficients c_j depend on q and d (recall that either $b = 0$ or $b = d$). For future reference, it is convenient to specify the values of r and q for the various types of methods. In Table 3.1 these values are listed for general values of d.

For an arbitrary given value of d, the order of consistency of the stability function (3.5a) cannot exceed r, hence, by choosing m such that the order p of the PDIRK method equals r, we achieve that the number of sequential stages is minimal with respect to the order p.

TABLE 3.1
Values of r and q in the stability function (3.5a)

Type	IA	IB	IC	IIA	IIB	IIC
$r =$	$m+1$	$m+1$	$m+2$	m	m	$m+1$
$q =$	m	$m+1$	$m+1$	m	$m+1$	$m+1$

3.1.1. *Derivation of A-acceptable and L-acceptable stability functions.* The following theorem defines an explicit representation of the stability function.

Theorem 3.1. *Let p be the order of the method, and let m be such that $r = p$; then the coefficients of* (3.5a) *are given by*

$$c_j = \sum_{i=0}^{j} \binom{q}{j-i} \frac{(-1)^{j-i}}{i!d^i}, \qquad j = 0, 1, \ldots, q;$$

(3.5b)

$$c_j = \sum_{i=0}^{q} \binom{q}{i} \frac{(-1)^i}{(j-i)!d^{j-i}}, \qquad j = q+1, q+2, \ldots, p,$$

where $0! := 1$.

Proof. Since it is assumed that the method is of order p, we necessarily have $R(z) = \exp(z) + O(z^{p+1})$. By expanding the function $(1 - dz)^q \exp(z)$ in a Taylor series at $z = 0$, and by equating corresponding coefficients in this expansion and in the polynomial $P(z)$, defined in (3.5a), we can find the first $p+1$ coefficients of P. Hence, all coefficients of P are uniquely determined and are given by (3.5b) (see also Nørsett [19] and Butcher [3, p. 246] for expressions in terms of derivatives of Laguerre polynomials). □

Notice that the condition $r = p$ excludes methods of Type C.1, because for Type I and Type II variants the maximal order is $m+1$ and m, respectively, which is one lower than the corresponding value of r. As a consequence, for methods of Type C with stability functions of the form (3.5), the order should be increased by one, which is obtained by requiring the matrix E to satisfy the condition $E\mathbf{e} = A\mathbf{e} - d\mathbf{e}$.

By means of Theorem 3.1 the stability analysis is now rather straightforward. Following Nørsett [20] and Butcher [3], we write $u = y^2$ and define the so-called E-polynomial

$$\begin{aligned} E(u) &:= |(1 - iy)^q|^2[1 - |R(iy/d)|^2] = |(1 - iy)^q|^2 - |P(iy)|^2 \\ &= (1+u)^q - [c_0 - c_2u + c_4u^2 - \cdots]^2 - u[c_1 - c_3u + c_5u^2 - \cdots]^2. \end{aligned}$$

From the condition $R(z) = \exp(z) + O(z^{p+1})$ it follows that $|R(iy/d)|^2 = 1 + O(y^{p+1})$, so that $E(y^2) = O(y^{p+1})$. Hence, all terms of $E(y^2)$ of degree less than $p+1$ in y vanish, so that

$$E(u) = \sum_{j=[p/2]+1}^{q} e_j u^j, \qquad e_j = e_j(d) := \binom{q}{j} - c_j^2 - 2\sum_{i=1}(-1)^i c_{j-i}c_{j+i},$$

with $c_j := 0$ if $j > p$ or $j < 0$. Because of the maximum principle, we have

A-stability if $|R(iy)|$ is bounded by 1 for all real y, so that the method is A-stable if and only if $E(u)$ is nonnegative for $u \geq 0$.

Values of d for which $R(z)$ is A-acceptable will be called A-acceptable. Let the range of d-values which are A-acceptable be denoted by I_{pq}, i.e., $I_{pq} := \{d : E(u) \geq 0 \text{ for all } u \geq 0\}$; then the following summary is easily obtained by using Table 3.1 and the order results obtained for the various types of methods (p^* denotes the order of the corrector $\{A, \mathbf{b}\}$):

TABLE 3.2
Summary of properties of PDIRK methods with constant diagonal elements

Type	Condition	Order	Sequential stages	A-acceptable d-values
IA.1	$m \leq p^* - 1$	$m+1$	m	$I_{m+1,m}$
IB.1	$m \leq p^* - 1$	$m+1$	$m+1$	$I_{m+1,m+1}$
IC.2	$m \leq p^* - 2$	$m+2$	$m+1$	$I_{m+2,m+1}$
IIA.1	$m \leq p^*$	m	m	$I_{m,m}$
IIB.1	$m \leq p^*$	m	$m+1$	$I_{m,m+1}$
IIC.2	$m \leq p^* - 1$	$m+1$	$m+1$	$I_{m+1,m+1}$

Notice that $R(z)$ is L-acceptable if $R(z)$ is A-acceptable and if $q > p$. From Table 3.2 we see that the methods of Type IIB.1 possess L-acceptable stability functions. Since L-stable methods are usually more suitable for integrating stiff equations than A-stable methods, the methods of Type IIB.1 are of interest in spite of the additional sequential stage when compared with the other methods. However, just as in the case of SDIRK methods, it is possible that an A-stable method can be made L-stable if the interval of A-acceptable d-values contains a value for which c_p vanishes. For $q = p \leq 15$, this has been investigated by Wolfbrandt [25] and it was found that such values of d exist for $p \leq 6$ and $p = 8$. This information is summarized in Table 3.3a.

In a similar way, L-acceptable ranges of d-values can be found in the case $q = p + 1$. These ranges turn out to be nonempty for $p \leq 8$ and for $p = 10$, and are given in Table 3.3b. Moreover, we list the values of $d_{p,p+1}$, which are inside these L-acceptable ranges and cause c_p to vanish, resulting in even stronger damping at "infinity" (L^2-stability).

Finally, we considered the case $q = p - 1$, resulting from IA.1 and IC.2 type methods. Since now the degree of the numerator in $R(z)$ is larger than that of the denominator, a necessary condition for this case to yield A-stability is that c_p vanishes. For $p = 2, 3, \ldots, 10$ we determined the zeros of $c_p(d)$ and checked the resulting stability function on A-acceptability. Only for $p = 2$ ($d = \frac{1}{2}$), $p = 3$ ($d = (3+\sqrt{3})/6$), and $p = 4$ ($d = 1.0685790213$) A-stability can be obtained. Hence, in this way we have found A-stable methods of orders p up to 4 requiring $p - 1$ sequential stages. This result is similar to what is possible in the case of RK methods for sequential computers (cf. [1]); however, the present methods contain embedded formulas of lower order.

TABLE 3.3a
A-acceptable and L-acceptable values of d for $p = q$

$p = q$	Range I_{pp}	d_{pp}
1	$[1/2, \infty]$	1
2	$[1/4, \infty]$	$1 \pm \sqrt{1/2}$
3	[1/3, 1.068]	0.43586650
4	[0.395, 1.280]	0.5728160625
5	[0.247, 0.361] + [0.421, 0.473]	0.2780538410
6	[0.285, 0.54]	0.3341423671
7	empty	
8	[0.218, 0.264]	0.2343731596
9	empty	
10	empty	

TABLE 3.3b
Ranges of L-acceptable values of d for $p = q - 1$

$p = q - 1$	Range $I_{p,p+1}$	$d_{p,p+1}$
1	$[1 - \sqrt{1/2}, 1 + \sqrt{1/2}]$	0.5
2	[0.181, 2.185]	$0.5 \pm \sqrt{1/12}$
3	[0.224, 0.572]	0.3025345782
4	[0.248, 0.676]	0.3888576711
5	[0.184, 0.334]	0.2168805435
6	[0.205, 0.378]	0.2579552416
7	[0.157, 0.2029] + [0.2052, 0.234]	0.1690246379
8	[0.171, 0.259]	0.1929778040
9	empty	
10	[0.147, 0.165] + [0.1938, 0.1961]	0.1541460739

Notice that any s-stage, pth-order corrector (even explicit corrector methods) can be used for generating A-stable methods of Type IB, and any pth-order corrector satisfying the condition $\mathbf{b}^T = \mathbf{e}_s^T A$ for generating the A-stable methods of Type IIA and IIC, or the L-stable methods of Type IIB.

Furthermore, we have seen that the stability can be improved by selecting special d-values. Another possibility, which might be useful in a variable-stepsize implementation, is to exploit the *length* of the A- and L-acceptable ranges: for small changes in the stepsize h, the value of hd could be kept fixed (as long as the corresponding d-value is still in the allowed range, of course), so that a new decomposition of $I - hd\partial f/\partial y$ can be avoided.

3.1.2. *Accuracy test.* It is well known [7] that, when integrating general stiff systems, the actually observed order is usually much lower than the *classical order* p. In fact, the order behavior is often dictated by the so-called *stage-order* r (for a definition of this notion and its consequences the reader is referred to [7]). Since most (P)DIRK methods have stage order $r = 1$, one might question the relevance of PDIRK methods possessing a high classical order. And indeed, for a general stiff problem, this order-reduction phenomenon has great impact on the accuracy of this type of methods.

However, in [10], Hairer et al. give a thorough analysis of the behavior of RK methods when applied to a singularly perturbed problem of the form

$$\varepsilon \frac{dy_1}{dt} = f_1(y_1, y_2), \quad \frac{dy_2}{dt} = f_2(y_1, y_2), \quad \text{with } \varepsilon << 1, \tag{3.6}$$

and show that for special RK methods the classical order may still dominate the global error, especially if stiffness increases (i.e., if $\varepsilon \to 0$). The motivation for considering this particular problem class is that it has practical significance and has been extensively studied in the literature (see the references cited in [10]). An important characteristic of problems of the form (3.6) is that the eigenvalues of the Jacobian matrix can be clustered into two groups, and behave as $O(1)$ and $O(\varepsilon^{-1})$, respectively. Here we give the essential result of Hairer, Lubich, and Roche concerning the global error (cf. [10, Theorem 1 on p. 680]):

Theorem 3.2. *Let the RK method be A-stable and let* $\varepsilon \leq \text{Constant} \cdot h$. *Then the global error for the stiff component* y_1 *behaves as* $O(\varepsilon h^r) + O(h^p)$ *if* $\mathbf{b}^T = \mathbf{e}_s^T A$ *and as* $O(h^{r+1})$ *if* $|R(\infty)| < 1$. *For both cases, the global error for the nonstiff component* y_2 *behaves as* $O(\varepsilon h^{r+1}) + O(h^p)$.

This result indicates that Type II methods are to be preferred if $\varepsilon \to 0$, since then the global error is dominated by the classical order, whereas methods of Type I will behave according to their (low) stage order.

To illustrate these properties, we applied a few of the PDIRK methods derived in the preceding subsection to a problem of the form (3.6), proposed by Kaps [17]:

$$\begin{aligned} &\frac{dy_1}{dt} = -(2 + \varepsilon^{-1})y_1 + \varepsilon^{-1}(y_2)^2, \qquad \frac{dy_2}{dt} = y_1 - y_2(1 + y_2), \\ &y_1(0) = y_2(0) = 1, \quad 0 \leq t \leq 1, \end{aligned} \tag{3.6'}$$

with the smooth exact solution $y_1 = \exp(-2t)$ and $y_2 = \exp(-t)$ for all values of the parameter ε.

The methods we have used in our tests are based on correctors of different classical order (a specification of these correctors can be found in the appendix to the report [14]). Moreover, all methods were equipped with the special d_{pp} or $d_{p,p+1}$ values given in Table 3.3 and, consequently, are L-stable and L^2-stable, respectively.

TABLE 3.4
Values of Δ *at* $t = 1$ *for the first component of problem* (3.6′) *with* $\varepsilon = 10^{-8}$

Type	Corrector	Order	$h = \frac{1}{4}$	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$h = \frac{1}{32}$	$h = \frac{1}{64}$	Seq. Stages per step	Number of Processors
IB.1	Radau IIA	3	3.7	4.1	4.6	5.2	5.8	3	2
	Gauss-Legendre	4	2.9	3.6	4.2	4.8	5.4	4	2
	Explicit RK	4	3.0	3.7	4.3	4.9	5.5	4	4
	Radau IIA	5	3.6	4.3	4.9	5.5	6.1	5	3
	Gauss-Legendre	6	3.1	3.7	4.4	5.0	5.6	6	3
IIA.1	Radau IIA	3	4.0	4.9	5.8	6.7	7.6	3	2
	Radau IIA	5	6.9	8.4	9.8	10.6	11.0	5	3
IIB.1	Radau IIA	3	4.3	5.2	6.1	7.0	7.9	4	2
	Radau IIA	5	7.2	8.7	10.3	11.8	11.8	6	3
	Radau IIA	7	9.7	10.2	10.6	10.9	11.2	8	4
IIC.2	Radau IIA	3	4.0	4.9	5.8	6.7	7.6	3	2
	Radau IIA	5	6.9	8.4	9.8	10.6	11.0	5	3

For $\varepsilon = 10^{-8}$ the absolute error for the stiff component y_1 at the endpoint $t = 1$ is given in Table 3.4 (see p. 148); here, the error is written in the form $10^{-\Delta}$ and the values of Δ are listed. Notice that the Type II methods require a stiffly accurate corrector (such as the Radau IIA formulas) and that L-stable, seventh-order PDIRK methods are only possible within the family of Type IIB.1 methods (cf. Tables 3.2 and 3.3b). This table clearly demonstrates the superiority of the stiffly accurate Type II methods over the Type I methods, which show only a second-order behavior for the global error (recall that $r = 1$ for the Type IB.1 methods). On the other hand, the stiffly accurate methods exhibit the classical order in the error behavior and thus both results are in perfect agreement with the estimates in the theorem of Hairer et al.

From this experiment we may conclude that it is relevant indeed to have high-order PDIRK methods for integrating stiff systems of the form (3.6), in spite of their low stage order.

Comparing the efficiency of the various parallel methods of Type II, we observe that schemes of Types A and C are equally efficient, since they require the same number of sequential stages (cf. Table 3.2). Type IIB.1 methods yield slightly more accurate results, but need an additional stage to reach the same order (we remark that the seventh-order method of this type does not show full advantage, since the integration process was impeded by the machine precision).

3.2. **PDIRK methods with arbitrary diagonal matrices.** In the case where B and D are allowed to be arbitrary diagonal matrices, it is convenient to express $Q_m(z)$ in the form

$$\begin{aligned} Q_m(z) &= [I - Z]^{-1}[I - Z^m] + Z^m Q_0 \\ &= [I - Z]^{-1}[I - Z^m] + Z^m[I - zD][I - zB]^{-1}[I + zE]. \end{aligned}$$

Since $[I - zD]^{-1} = [I - zA]^{-1}[I - Z]$, we find

$$Q_m(z) = [I - zD][I - zA]^{-1}\left[I - Z^m + [I - Z]Z^m[I - zD][I - zB]^{-1}[I + zE]\right],$$

so that (3.3) yields

$$\begin{aligned} R_m^{\mathrm{I}}(z) &= 1 + z\mathbf{b}^{\mathrm{T}}[I - zA]^{-1}\left[I - Z^m + [I - Z]\,Z^m \right. \\ &\qquad \left. \times [I - zD][I - zB]^{-1}[I + zE]\right]\mathbf{e}, \\ R_m^{\mathrm{II}}(z) &= \mathbf{e}_s^{\mathrm{T}}[I - zA]^{-1}\left[I - Z^m + [I - Z]Z^m \right. \\ &\qquad \left. \times [I - zD][I - zB]^{-1}[I + zE]\right]\mathbf{e} \\ &= 1 + \mathbf{e}_s^{\mathrm{T}}[I - zA]^{-1}\left[zA - Z^m + [I - Z]Z^m \right. \\ &\qquad \left. \times [I - zD][I - zB]^{-1}[I + zE]\right]\mathbf{e}. \end{aligned} \tag{3.3'}$$

In the following two subsections, a representation for the stability functions without inverses of matrices will be given, and stability characteristics of PDIRK methods of Types IB.2, IIB.2, and IIC.3 are presented.

3.2.1. *Representation theorems.* The following theorem gives a representation of the stability functions in terms of determinants containing only inverses of diagonal matrices.

Theorem 3.3. *The stability functions* (3.3′) *can be represented by*

$$R_m^{\mathrm{I}}(z) = \frac{\det\{I-zA+z[I-Z^m+[I-Z]Z^m[i-zD][I-zB]^{-1}[I+zE]]\mathbf{e}\mathbf{b}^{\mathrm{T}}\}}{\det\{I-zA\}},$$
$$R_m^{\mathrm{II}}(z) = \frac{\det\{I-zA+[zA-Z^m+[I-Z]Z^m[I-zD][I-zB]^{-1}[I+zE]]\mathbf{e}\mathbf{e}_s^{\mathrm{T}}\}}{\det\{I-zA\}}. \tag{3.7}$$

Proof. Applying the identity

$$1+\mathbf{x}^{\mathrm{T}}N^{-1}\mathbf{y} = \frac{\det\{N+\mathbf{y}\mathbf{x}^{\mathrm{T}}\}}{\det\{N\}}$$

to the stability functions (3.3′) straightforwardly leads to the representations (3.7). □

The expressions (3.7) can be simplified for the respective Types A, B, and C:

Corollary 3.1. *Let the matrix* Z *be given by* $Z = z(A-D)(I-zD)^{-1}$; *then the following assertions hold*:

(a) *The stability function of PDIRK methods of Type* A.1 *are given by*

$$R_m^{\mathrm{I}}(z) = \frac{\det\{I-zA+z[I-zZ^mA]\mathbf{e}\mathbf{b}^{\mathrm{T}}\}}{\det\{I-zA\}},$$
$$R_m^{\mathrm{II}}(z) = \frac{\det\{I-zA+z[I-Z^m]A\mathbf{e}\mathbf{e}_s^{\mathrm{T}}\}}{\det\{I-zA\}}. \tag{3.8a}$$

(b) *The stability function of PDIRK methods of Type* B *are given by*

$$R_m^{\mathrm{I}}(z) = \frac{\det\{I-zA+z[I-Z^{m+1}]\mathbf{e}\mathbf{b}^{\mathrm{T}}\}}{\det\{I-zA\}},$$
$$R_m^{\mathrm{II}}(z) = \frac{\det\{I-zA+[zA-Z^{m+1}]\mathbf{e}\mathbf{e}_s^{\mathrm{T}}\}}{\det\{I-zA\}}. \tag{3.8b}$$

(c) *The stability function of PDIRK methods of Type* C.2 *or Type* C.3 *are given by*

$$R_m^{\mathrm{I}}(z) = \frac{\det\{I-zA+z[I-zZ^{m+1}A]\mathbf{e}\mathbf{b}^{\mathrm{T}}\}}{\det\{I-zA\}},$$
$$R_m^{\mathrm{II}}(z) = \frac{\det\{I-zA+z[I-Z^{m+1}]A\mathbf{e}\mathbf{e}_s^{\mathrm{T}}\}}{\det\{I-zA\}}. \tag{3.8c}$$

Notice that these expressions no longer explicitly depend on E and B, and are completely determined by the corrector and the matrix Z.

3.2.2. *Stability characteristics.* In this subsection, we consider the stability of PDIRK methods. We shall distinguish between methods based on Radau IIA correctors and on Gauss-Legendre correctors.

The Radau IIA correctors have order $p = 2s-1$, where s is the number of stages, and satisfy the condition $\mathbf{b}^{\mathrm{T}} = \mathbf{e}_s^{\mathrm{T}}A$ (their Butcher arrays for $s = 1, \ldots, 4$ are given in the appendix to [14]). Owing to this property, PDIRK methods of Type I and Type II are both relevant. We confine our considerations to types which require (with respect to their order) less sequential stages than the corresponding methods indicated in Table 3.2, that is, we consider methods of Types IB.2, IIB.2, and IIC.3. For these types of methods, the stability

functions are completely determined. In Table 3.5, we present a summary of the characteristics of these methods for several orders. Based on the stability functions (3.8), the stability region of the methods was determined numerically. It turned out that some stability functions are only $A(\alpha)$-acceptable. However, in these cases α is very close to 90° (in the Appendix to [14], a set of stability regions is given, including the regions of the embedded lower-order methods).

Furthermore, we considered PDIRK methods based on Gauss-Legendre correctors. Such s-stage correctors have order $2s$, but are not stiffly accurate and, hence, only Type I methods are relevant. In Table 3.5 we have included the characteristics of fourth- and sixth-order methods of Type IB.2 (the generating correctors can be found in [3, p. 219]).

TABLE 3.5
Characteristics of PDIRK *methods based on arbitrary B and D matrices*

Type	Corrector	Order	Seq. Stages	Proces-sors	Stability
IB.2	Radau IIA	3	2	2	Strongly A-stable
	Gauss-Legendre	4	3	2	Strongly A-stable
	Radau IIA	5	4	3	Strongly A-stable
	Gauss-Legendre	6	5	3	Strongly $A(\alpha)$-stable, $\alpha = 89.97°$
	Radau-IIA	7	6	4	Strongly $A(\alpha)$-stable, $\alpha = 83.3°$
IIB.2	Radau IIA	3	3	2	$L(\alpha)$-stable, $\alpha = 89.75°$
	Radau IIA	5	5	3	$L(\alpha)$-stable, $\alpha = 89.12°$
	Radau IIA	7	7	4	$L(\alpha)$-stable, $\alpha = 89.02°$
IIC.3	Radau IIA	3	2	2	A-stable
	Radau IIA	5	4	3	$A(\alpha)$-stable, $\alpha = 89.997°$
	Radau IIA	7	6	4	$A(\alpha)$-stable, $\alpha = 89.95°$

In comparison with the PDIRK methods constructed in §3.1, we observe that the above PDIRK methods of Types IB.2 and IIC.3 require one sequential stage less to obtain a given order of accuracy. Moreover, with the exception of the seventh-order method of Type IB.2, these methods possess almost the same good stability properties.

For the methods of Type IIB.2 (for which the order equals the number of sequential stages), only the seventh-order is relevant, since in §3.1 it turned out to be impossible to construct an L-stable method of order 7 with seven sequential stages; the third- and fifth-order methods of Type IIB.2 do not have an advantage over the L-stable methods described in §3.1.

3.2.3. *Accuracy test.* We conclude this section by applying the methods specified in Table 3.5 to the problem (3.6′). Using the same notation as described in §3.1.3, we give the results in Table 3.6 (see p. 152).

TABLE 3.6
Values of Δ *at* $t = 1$ *for the first component of problem* (3.6′) *with* $\varepsilon = 10^{-8}$

Type	Corrector	Order	$h = \frac{1}{4}$	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$h = \frac{1}{32}$	$h = \frac{1}{64}$	Seq. Stages per step	Number of Processors
IB.2	Radau IIA	3	2.8	3.8	4.1	4.7	5.3	2	2
	Gauss-Legendre	4	2.7	3.4	4.0	4.6	5.3	3	2
	Radau IIA	5	2.4	2.8	3.4	4.1	4.8	4	3
	Gauss-Legendre	6	3.0	3.5	4.1	4.8	5.4	5	3
	Radau IIA	7	4.2	4.6	5.2	5.8	6.4	6	4
IIB.2	Radau IIA	3	3.4	4.1	4.9	5.8	6.7	3	2
	Radau IIA	5	4.9	6.1	7.5	9.0	10.4	5	3
	Radau IIA	7	6.4	8.2	10.1	11.9	12.5	7	4
IIC.3	Radau IIA	3	4.3	5.2	6.1	7.0	7.9	2	2
	Radau IIA	5	6.6	8.0	9.4	10.8	11.6	4	3
	Radau IIA	7	8.7	10.6	12.0	12.3	12.6	6	4

Again, the stiffly accurate Type II methods are much more efficient than the methods of Type I. Moreover, the order behavior nicely illustrates the results of the theorem of Hairer et al. (cf. §3.1.2). Furthermore, within the class of stiffly accurate methods, the C-variant is superior to the B-variant, since it is cheaper and yields, for this example, more accuracy.

4. Efficiency Tests

Finally, we investigate the performance of PDIRK methods when run on a parallel computer. Because it is highly desirable to use an unconditionally stable method of high order, we selected a PDIRK method of Type IIB.1 with a D-matrix of the form $D = dI$. On the basis of the accuracy test described in §3.1.2, we decided to choose the seventh-order, four-point Radau IIA corrector (see (A.3) in the Appendix to [14]), with $m = 7$ iterations. The resulting method is of order seven (cf. Theorem 2.2), and by choosing $d = 0.1690246379$ we achieve strong damping at infinity (L^2-stability, cf. Table 3.3b). Hence, taking into account the (implicit) predictor, the method requires eight sequential stages per step. We have implemented this method on an Alliant FX/4 computer having four parallel (vector-) processors, shared memory, and approximately 16-digit arithmetic precision. Since the underlying Radau method has four stages, we may expect an efficient use of this machine.

In order to be able to test problems with a strongly fluctuating solution, we equipped the above fixed-order PDIRK method with a simple strategy for error control and stepsize selection. Since the PDIRK approach provides a whole set of embedded reference solutions of lower order, we can construct an estimate of the local truncation error without additional costs. For this purpose we take $\|\mathbf{e}_4^T\mathbf{Y}^{(m)} - \mathbf{e}_4^T\mathbf{Y}^{(m-1)}\|$ as an estimate for the local error. All implicit relations are iterated using modified Newton iteration. If convergence happens to fail within a fixed number of iterations (in our version, we choose this number equal to 10), then we update the Jacobian and, if still no convergence can be obtained, we halve the stepsize (repeatedly, if necessary). Furthermore, the Newton process to solve for $\mathbf{Y}^{(j)}$ is started with the initial guess $\mathbf{Y}^{(j-1)}$, which is of increasing accuracy for increasing j. It should be observed that this provisional implementation certainly can be improved by a better tuning of the separate elements (for example, all kinds of thresholds and strategy parameters should be tuned on the basis of extensive testing). Since it is not the aim of this paper to present such a "production code," we will give results for our "research version."

The goal of our tests is twofold:

(i) We want to investigate (for realistic problems) to what extent the theoretical parallelization can be realized in practice; in other words, what speedup factor can be obtained on this four-processor machine. Obviously, the ideal factor of four will be too optimistic because of unavoidable overhead, like communication and sequential parts in the program.

(ii) We want to compare the performance of the parallelized PDIRK code with that of a good sequential ODE solver. Within the class of sequential solvers based on unconditionally stable methods, we selected the code SIMPLE of Nørsett and Thomsen [22]. The method underlying this robust and reliable code is closely related to the PDIRK method, i.e., it is also based on an unconditionally stable, diagonally implicit Runge-Kutta method. Furthermore, SIMPLE is, like PDIRK, equipped with embedding techniques to control the

local error. A disadvantage of this code is that its order is rather low; it is based on a third-order DIRK method. However, high-order A-stable DIRK-codes are not available in the literature. Since many problems are more efficiently integrated if high-order formulas are available, we also looked for a code based on methods of various orders. This led us to LSODE of Hindmarsh [11]. This BDF-based code has enjoyed very successful usage over a long period. However, the fact that only the first- and second-order formula in this code are unconditionally stable makes LSODE less robust as a *general* stiff solver. It is well known that the performance of this code may deteriorate significantly when it is applied to problems with eigenvalues in the vicinity of the imaginary axis (see, for example, Stewart [23]). On the other hand, since LSODE is generally accepted as being a good sequential ODE solver, we decided to include it in our tests.

In the next subsections, we describe the results obtained when the aforementioned three codes are applied to some hard problems. Since the codes are different in nature (low order versus high order, one-step versus multistep), we refrain from indicating the traditional statistical output of an automatic ODE solver, like number of steps, number of LU decompositions, etc. It should be observed that the work involved per step is quite different for the various codes: for instance, the sequential number of implicit relations to be solved per step equals one for LSODE, four for SIMPLE, and eight for PDIRK. Since the codes do not yield equal accuracy for the same value of the local error control parameter TOL, we list results for various values of TOL and measure the accuracy produced as well as the CPU-time required. All accuracies are given in terms of Δ, the number of correct digits at the endpoint of the integration interval (see §3.1.2), and the CPU-times are given in seconds.

4.1. **Robertson kinetics example.** In our first example we solve a set of reaction-rate equations:

$$\begin{aligned} \frac{dy_1}{dt} &= -0.04y_1 + 10^4 y_2 y_3, \\ \frac{dy_2}{dt} &= 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 (y_2)^2, \\ \frac{dy_3}{dt} &= 3 \cdot 10^7 (y_2)^2, \end{aligned} \tag{4.1}$$

defined on the interval $[0, 10^8]$ with initial conditions $y_1(0) = 1$, $y_2(0) = y_3(0) = 0$. This problem is also used by Hindmarsh and Nørsett & Thomsen to illustrate the performance of LSODE and SIMPLE.

Initially, the solution changes rapidly, and small stepsizes are required; gradually the solution reaches a steady state, and the stepsize can be increased considerably. In a typical situation we observed stepsizes in the range $[10^{-3}, 10^6]$. Hence, this problem imposes a severe test on the stepsize selection procedure. The results obtained by the various codes are collected in Table 4.1. Here T_1 means the CPU-time when only one processor is used, and T_4 denotes the CPU-time required when the program is run on four processors.

These results give rise to the following conclusions:

(i) Concerning the parallelization of the PDIRK code we observe a speedup by a factor $(T_1/T_4 \approx)$ 2.68 and 2.91 for the two values of TOL that we have used. One reason why these numbers are less than the optimal speedup factor 4

TABLE 4.1
Δ-values and CPU-times for problem (4.1)

Method	TOL	Δ	T_1	T_4
	10^{-4}	6.5	0.63	0.85
SIMPLE	10^{-5}	7.8	1.38	$> T_1$
	10^{-6}	9.5	3.67	$> T_1$
	10^{-5}	7.4	0.35	$> T_1$
LSODE	10^{-7}	8.6	0.80	$> T_1$
	10^{-9}	10.3	1.71	$> T_1$
PDIRK	10^{2}	8.5	0.51	0.19
	10^{0}	11.1	1.08	0.37

is the introduction of inevitable overhead (and of scalar code). Another reason is algorithmic in nature. Each component of the prediction $\mathbf{Y}^{(0)}$ is a numerical approximation to the ODE solution at the point $t_n + dh$ (actually, all processors have solved exactly the same implicit relation in this predictor stage). These components are used as an initial guess in the various Newton processes computing $\mathbf{Y}^{(1)}$. Since the components of $\mathbf{Y}^{(1)}$ are approximations to the ODE solution at *different* points (i.e., the Radau points), these initial guesses do not have equal accuracy, so that we may expect different numbers of Newton iterations on the various processors. In the case TOL = 1, we measured the actual numbers of Newton iterations over the whole integration interval and found, for the four processors, 848, 924, 1012, and 1043, respectively. This means that in some steps a few processors have met the convergence criterion in the Newton process, and thus have been idle for some time while waiting for the other processors to complete solving their implicit relation. Taking this aspect into account, the optimal parallelization cannot exceed a speedup factor equal to $(848 + 924 + 1012 + 1043)/1043 \approx 3.67$. The measured speedup in this case equals 2.91 (i.e., 79%), showing that the overhead (communication, scalar code, etc.) only slightly degrades the performance. The reduction of the ideal factor 4 to 3.67 is a price we have to pay for choosing a PDIRK method. We may conclude that the actual efficiency of the method as a whole, defined as the total speedup divided by the number of processors used, equals $2.91/4 = 0.73$.

(ii) Concerning the scalar codes SIMPLE and LSODE, we observe that they run faster on one processor than on four (see the result obtained by SIMPLE for TOL = 10^{-4}). Apparently, the parallelization and vectorization overhead does not pay for this problem (this might be different in case of an ODE with many components). Therefore, we only give timings for the uniprocessor experiments.

(iii) When compared with PDIRK, we see that SIMPLE needs much more time in the high-accuracy range. This is obviously due to its low order. LSODE, which can utilize higher orders, is more efficient in this range but, when compared to PDIRK, its CPU-time is approximately four times larger to obtain 8.5 digits precision, and this factor increases if still higher-precision results are requested (notice that even on one processor, PDIRK is faster than LSODE on this problem).

(iv) Finally, we observe that the value for TOL used by PDIRK is several orders of magnitude larger than the value used by either SIMPLE or LSODE to achieve the same global error. This can be explained as follows: Owing to its high order, the local truncation error of PDIRK is usually relatively small. Therefore, if crude tolerances are used, the error control mechanism signals that a large stepsize can be used in order to balance the estimated and the requested local error. On the other hand, the Newton process imposes a limitation on the stepsize. In our implementation, the Newton processes to solve for $\mathbf{Y}^{(0)}$ are given the value y_n as initial iterate. Unfortunately, for large values of h (as suggested by the error estimator) this initial iterate is not always inside the contraction domain for the Newton process, resulting in an adequate reduction of the stepsize. As a consequence, this high-order scheme, using a small(er) stepsize, will produce a local error which is much smaller than requested.

In conclusion, for this test problem (and also for the problem to be discussed in the next subsection), the restriction on the stepsize imposed by the Newton process is more stringent than that imposed by the local error control, unless very small values for TOL are used. We have also integrated some *linear* ODE's (for which the convergence problems are not relevant, of course) and observed a relation between TOL and the global error similar to that of SIMPLE and LSODE.

Summarizing, for obtaining highly accurate results, the above experiment shows that the high order of the PDIRK method is worth the large amount of redundancy introduced in its construction. In this connection we remark that the order of these methods can still be raised to 10, whereas an increase of the order is not possible for BDF methods and not feasible for embedded DIRK methods underlying the SIMPLE code.

4.2. **Van der Pol's equation.** Our second example is given by the van der Pol equation

$$y'' - \mu(1 - y^2)y' + y = 0. \tag{4.2}$$

For $\mu = 5$, this is problem E2 from the test set of Enright et al. [8]. However, as reported there, on the interval $[0, 1]$ the spectral radius of the Jacobian does not exceed 15, so that the problem is not really stiff. Therefore, we set this parameter equal to 50. For this μ-value the equation exhibits so-called "relaxation oscillations," which means that the solution possesses internal boundary layers. Furthermore, we select an integration interval sufficiently large to capture such an internal layer, which again requires an adequate stepsize selection procedure. The problem tested in this section is defined by

$$\begin{aligned} \frac{dy_1}{dt} &= y_2, \qquad y_1(0) = 2, \\ \frac{dy_2}{dt} &= 50(1 - (y_1)^2)y_2 - y_1, \qquad y_2(0) = 0, \end{aligned} \qquad 0 \le t \le 41.5. \tag{4.3}$$

TABLE 4.2
Δ-values and CPU-times for problem (4.2)

Method	TOL	Δ	T_1	T_4
	10^{-6}	5.6	1.07	$> T_1$
SIMPLE	10^{-8}	6.9	5.64	$> T_1$
	10^{-10}	7.8	25.5	$> T_1$
	10^{-6}	4.3	0.24	$> T_1$
LSODE	10^{-8}	6.3	0.42	$> T_1$
	10^{-10}	7.8	0.83	$> T_1$
	10	5.1	0.56	0.20
PDIRK	10^{-2}	6.1	1.20	0.41
	10^{-5}	7.2	2.44	0.82

This test example has also been discussed by Gottwald and Wanner in [9]. At approximately $t = 40.7$, the solution y_1 drops from 1 to -2 on a very short interval, forcing the codes to reduce their steplengths dramatically (several orders of magnitude) The results of the various codes applied to this problem are given in Table 4.2.

Again, we see that PDIRK can take advantage from the availability of four processors: on the average, the speedup is 2.9 (or, equivalently, the efficiency is ≈ 0.72). For this problem, the loss in efficiency due to overhead is less than $(1 - 0.72 =)$ 0.28, because the various processors required a different number of Newton iterations (viz., for TOL $= 10^{-5}$ we found 3186, 3561, 3882, and 4092 iterations, respectively, which reduces the optimal speedup factor from 4 to 3.6).

Furthermore, it is quite clear that the low-order SIMPLE code becomes excessively more expensive for smaller values of TOL. On the other hand, LSODE behaves rather efficiently for this problem and is approximately equally efficient as PDIRK.

4.3. **Conclusions.** On the basis of these (difficult) problems we may draw the following conclusions:

- The actually obtained degree of parallelization of the PDIRK method is fairly close to its ideal value.
- The reason that SIMPLE is less efficient than the other two codes, especially in the high-accuracy range, is because of its low order.
- It is well known that the higher-order BDF formulas lack the property of L-stability. This may result in serious difficulties for LSODE in the case that the Jacobian has eigenvalues in the vicinity of the imaginary axis. However, the

two test problems do not belong to this category; hence, LSODE has not been faced with the limitation of the stability regions of the higher-order BDF's.

– Unlike the implementation of SIMPLE and LSODE, the implementation of PDIRK does not require additional costs in calculating a reference solution.

– The present research version of the PDIRK code is at least as efficient as the well-balanced, extensively tested LSODE code.

– A future version of a PDIRK code can be improved as follows:

(i) better tuning of the stepsize strategy parameters and, particularly, finding more accurate initial iterates for the Newton process in the prediction stage;

(ii) implementation of a variable-order strategy; L-stable PDIRK formulas of orders up to 10 (excluding order 9) are available;

(iii) implementation of a stiffness detector, like the one in SIMPLE, and switching to parallel fixed-point iteration (PIRK methods, cf. [12]) in nonstiff regions of the integration interval.

ACKNOWLEDGMENT

The authors would like to thank Dr. W. Hundsdorfer for the fruitful discussions on the order-reduction phenomenon and W. Lioen for assisting them with the experiments on the Alliant FX/4.

BIBLIOGRAPHY

1. R. Alexander, *Diagonally implicit Runge-Kutta methods for stiff ODE's*, SIAM J. Numer. Anal. **14** (1977), 1006–1021.
2. K. Burrage, *The error behaviour of a general class of predictor-corrector methods*, CMSR Report, University of Liverpool, 1989.
3. J. C. Butcher, *The numerical analysis of ordinary differential equations, Runge-Kutta and general linear methods*, Wiley, New York, 1987.
4. J. R. Cash, *Diagonally implicit Runge-Kutta formulae with error estimates*, J. Inst. Math. Appl. **24** (1979), 293–301.
5. J. R. Cash and C. B. Liem, *On the design of a variable order, variable step diagonally implicit Runge-Kutta algorithm*, J. Inst. Math. Appl. **26** (1980), 87–91.
6. M. Crouzeix, *Sur l'approximation des équations différentielles opérationnelles linéaires par des méthodes de Runge-Kutta*, Ph. D. Thesis, Université de Paris, 1975.
7. K. Dekker and J. G. Verwer, *Stability of Runge-Kutta methods for stiff nonlinear differential equations*, CWI Monograph, no. 2, North-Holland, Amsterdam-New York-Oxford, 1984.
8. W. H. Enright, T. E. Hull, and B. Lindberg, *Comparing numerical methods for stiff systems of ODEs*, BIT **15** (1975), 10–48.
9. B. A. Gottwald and G. Wanner, *A reliable Rosenbrock integrator for stiff differential equations*, Computing **26** (1981), 355–360.
10. E. Hairer, Ch. Lubich, and M. Roche, *Error of Runge-Kutta methods for stiff problems studied via differential algebraic equations*, BIT **28** (1988), 678–700.
11. A. C. Hindmarsh, *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM/SIGNUM Newsletter (4) **15** (1980), 10–11.
12. P. J. van der Houwen and B. P. Sommeijer, *Variable step iteration of high-order Runge-Kutta methods on parallel computers*, J. Comp. Appl. Math. **29** (1990), 111–127.
13. ——, *Iterated Runge-Kutta methods on parallel computers*, SIAM J. Sci. Statist. Comput. **12** (1991), 1000–1028.

14. P. J. van der Houwen, B. P. Sommeijer, and W. Couzy, *Embedded diagonally implicit Runge-Kutta algorithms on parallel computers*, Report NM-R8912, Centre for Mathematics and Computer Science, Amsterdam, 1989.
15. A. Iserles and S. P. Nørsett, *On the theory of parallel Runge-Kutta methods*, IMA J. Numer. Anal. **10** (1990), 463–488.
16. K. Jackson and S. P. Nørsett, *Parallel Runge-Kutta methods*, manuscript, 1988.
17. P. Kaps, *Rosenbrock-type methods*, Numerical Methods for Stiff Initial Value Problems (G. Dahlquist and R. Jeltsch, eds.), Bericht Nr. 9, Inst. für Geometrie und Praktische Mathematik der RWTH Aachen, 1981.
18. I. Lie, *Some aspects of parallel Runge-Kutta methods*, Report No. 3/87, Division of Numerical Mathematics, University of Trondheim, 1987.
19. S. P. Nørsett, *Semi-explicit Runge-Kutta methods*, Report Mathematics and Computation No. 6/74, Dept. of Mathematics, University of Trondheim, 1974.
20. ——, *C-polynomials for rational approximation to the exponential function*, Numer. Math. **25** (1975), 39–56.
21. S. P. Nørsett and H. H. Simonsen, *Aspects of parallel Runge-Kutta methods*, Numerical Methods for Ordinary Differential Equations (A. Bellen, C. W. Gear, and E. Russo, eds.), Proceedings L'Aquila 1987, Lecture Notes in Math., vol. 1386, Springer-Verlag, Berlin, 1989, pp. 103–117.
22. S. P. Nørsett and P. G. Thomsen, *Embedded SDIRK-methods of basic order three*, BIT **24** (1984), 634–646.
23. K. Stewart, *Avoiding stability-induced inefficiencies in BDF methods*, J. Comput. Appl. Math. **29** (1990), 357–367.
24. H. W. Tam, *Parallel methods for the numerical solution of ordinary differential equations*, Report NO. UIUCDCS-R-89-1516, Computer Science Department, University of Illinois, 1989.
25. A. Wolfbrandt, *A study of Rosenbrock processes with respect to order conditions and stiff stability*, Ph. D. Thesis, Chalmers University of Technology, Göteborg, 1977.

CENTRE FOR MATHEMATICS AND COMPUTER SCIENCE, P. O. BOX 4079, 1009 AB AMSTERDAM, THE NETHERLANDS