Parallel Runge-Kutta-Nyström Methods

Parallel Runge-Kutta-Nyström Methods

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof. dr. P. W. M. de Meijer
in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui)
op dinsdag 29 maart 1994 te 15.00 uur

door

Nguyen huu Cong

geboren te Xuan Hai, Nghi Xuan, Ha Tinh (Vietnam)

Parallel Runge-Kutta-Nyström methods

Promotor	: Prof. Dr. P. J. van der Houwen
Co-promotor	: Dr. B. P. Sommeijer
Faculteit	: Wiskunde en Informatica

Printed at the Centrum voor Wiskunde en Informatica (CWI), Amsterdam

To the memory of my Mother and Father

Preface

The research described in this thesis has been supported by the University of Amsterdam by means of a research grant in the framework of the VH25 project. This project is a cooperation between the University of Amsterdam and the University of Hanoi. It offered me the opportunity to spend a total of two years at the Centre for Mathematics and Computer Science (CWI) in Amsterdam. The research started in the summer of 1990 and has been carried out, alternately, in Amsterdam and Hanoi. The investigations have resulted in six papers, which have been published (or are about to be published) in scientific journals. Preceded by an introduction, these papers form the six chapters of this thesis. The first four chapters deal with the construction and analysis of (parallel) methods for special second-order differential equations, whereas the last two chapters discuss similar methods for first-order differential equations. The papers are:

[1] *Stability of collocation-based Runge-Kutta-Nyström methods*, BIT **31** (1991), 469-481.
(P. J. van der Houwen, B. P. Sommeijer and Nguyen huu Cong)

[2] *Parallel diagonally implicit Runge-Kutta-Nyström methods,* Appl. Numer. Math. **9** (1992), 111-131.
(P. J. van der Houwen, B. P. Sommeijer and Nguyen huu Cong)

[3] *A-stable diagonally implicit Runge-Kutta-Nyström methods for parallel computers*, Numerical Algorithms **4** (1993), 263-281.
(Nguyen huu Cong)

[4] *Note on the performance of direct and indirect Runge-Kutta-Nyström methods*, J. Comput. Appl. Math. **45** (1993), 347-355.
(Nguyen huu Cong)

[5] *Parallel block predictor-corrector methods of Runge-Kutta type,* Appl. Numer. Math. **13** (1993), 109-123.
(P. J. van der Houwen and Nguyen huu Cong)

[6] *Parallel iteration of symmetric Runge-Kutta methods for nonstiff problems,* to appear in J. Comput. Appl. Math. (1994).
(Nguyen huu Cong)

On completion of this thesis, I would like to express my gratitude to Prof. Dr. J. Th. Runnenburg and Prof. Dr. Hoang Huu Nhu, the coordinators of VH25, for offering me the opportunity to join the project. Especially, I wish to thank Prof. Runnenburg for all his organizational work involved with my stay in Amsterdam.

I am very much indebted to my promotor Prof. Dr. P. J. van der Houwen, whose inspiring guidance and critical help brought me to the successful completion of the thesis. My deep appreciation is also to my co-promotor Dr. B. P. Sommeijer for his stimulating help and for the many discussions we had.

Many thanks are expressed to the colleagues at CWI: Henk Boender, Joke Blom, Kees Everaars, Erik de Goede, Piet Hemker, Marije Huizing, Willem Hundsdorfer, Jan Kok, Barry Koren, Choi-Hong-Lai, Walter Lioen, Maarten van Loon, Margreet Louter-Nool, Herman te Riele, Ron Trompert, Jan Verwer, Dik Winter, Simone van der Wolff, and Paul de Zeeuw. They all contributed in their own way to this thesis. Special thanks go to Walter Lioen, Erik de Goede and Dik Winter for their friendship and for their help in acquainting me with the rapidly changing hardware and software.

I also like to thank my colleagues from the University of Hanoi: Prof. Dr. Pham Ky Anh, Prof. Dr. Tran Van Nhung, Dr. Pham Trong Quat, and Prof. Dr. Dao Trong Thi for their support and their encouragement.

I express my gratitude to the board of CWI for the hospitality and for providing me with all computing facilities, and to the publication department for printing this thesis.

I am grateful to Ir. P. de Goeje and Drs. M. Lamers from the *'bureau ontwikkelingssamenwerking'* for all their efforts related to my stay in Amsterdam.

Finally, I want to record my warm appreciation to my wife Hanh, my son Anh Cuong and my daughter Hanh Dung for keeping me always in a happy family.

Amsterdam, January 1994 Nguyen huu Cong

Contents

Introduction

The aim of this thesis is the construction and analysis of parallel integration techniques for solving the special second-order initial-value problem

$$\frac{d^2\mathbf{y}(t)}{dt^2} = \mathbf{f}(t, \mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}'(t_0) = \mathbf{y}'_0,$$

(1)

$$\mathbf{y} : \mathbb{R} \to \mathbb{R}^d, \qquad \mathbf{f} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d.$$

One (simple) option for solving this problem consists in writing the problem in first-order form and applying a parallel integration method for first-order differential equations, without taking into account the special form of problem (1) (the "indirect" approach). However, ignoring the fact that **f** does not contain the first derivative, usually leads to algorithms that are less efficient than algorithms tuned to the special form of (1) (the "direct" approach). We illustrate this by an example from the class of *sequential* Runge-Kutta type methods (a method will be called *sequential* if it is designed for sequential computers). The highest-order, explicit Runge-Kutta method for first-order equations available in the literature, is the 17-stage, tenth-order Runge-Kutta method of Hairer (1978). Thus, writing (1) in first-order form and applying Hairer's method requires 17 evaluations of **f** per integration step. Alternatively, we can pick a Runge-Kutta type method directly designed for problems like (1). Such methods are generally known as Runge-Kutta-Nyström (RKN) methods. In Hairer (1982), we can find an RKN method of order 10 requiring 11 evaluations of **f** per integration step. Hence, in this example, exploiting the special form of the differential equation, saves 6 evaluations of **f** per step.

The preceding example compares the direct and indirect approach for sequential methods. It is highly likely that in the class of parallel methods, the direct approach will also lead to an improvements of the efficiency. This motivated us to develop direct parallel methods for solving problem (1), rather than using existing parallel methods for first-order problems via the indirect approach. In this thesis we will show that parallel RKN methods do improve the efficiency substantially, not only with respect to the best sequential RKN methods, but also with respect to existing parallel RK integration techniques.

1. Parallel explicit Runge-Kutta-Nyström methods

Let us assume that the problem (1) is stable, that is, the Jacobian matrix of **f** is assumed to have its eigenvalues on the *negative* axis. In analogy with the

terminology used for first-order problems, we shall call (1) *nonstiff* if these eigenvalues are "not too far away " from the origin, say in the interval [-1, 0]. Such problems frequently arise in celestial mechanics. For nonstiff problems, we do not need *implicit* integration methods, so that we can confine our considerations to *explicit* RKN methods.

Following the approach used in Sommeijer (1992) for solving first-order equations, we have constructed parallel RKN methods by fixed point iteration of a suitable, usually implicit, RKN method (to be referred to as the *corrector method*). Fixed point iteration has a high degree of parallelism because the components of the stage vector iterates can all be computed in parallel. Hence, the sequential (or effective) costs of one iteration equal the costs of one evaluation of **f** and are independent of the number of stages in the corrector method. Here, sequential costs are understood to be the costs when the number of available processors equals the number of stages of the corrector method. If in each new step the iteration process is started with the numerical solution obtained in the preceding step, then fixed point iteration yields an explicit RKN method of order min{p, 2m+2} requiring s(m+1) evaluations of **f** per integration step. Here, m is the number of iterations, and p and s are the order and the number of stages of the underlying corrector method, respectively. However, the *sequential* number of stages of this explicit RKN method is only m+1. Hence, choosing m = [(p-1)/2] yields an explicit RKN method of order p requiring only [(p+1)/2] sequential evaluations of **f** per integration step, provided that s processors are available ([.] denotes the integer-part function). For example, taking the 10th-order RKN method of Hairer as the corrector method, we can construct an explicit, 10th-order RKN method with effectively only 5 evaluations of **f** per step. This would be twice as "cheap" as the aforementioned sequential RKN method of Hairer. Notice that in this example, the corrector method is *explicit*.

The preceding discussion indicates that many sequential RKN methods can be economized by fixed point iteration. However, since the number of necessary processors equals the number of stages of the corrector, we should take the number of stages of the corrector into account. For example, using the 10th-order RKN method of Hairer as corrector would require as many as 11 processors. Thus, our next step is to look for corrector methods possessing a small number of stages s relative to their order p. Such methods can be found in Hairer (1977, 1979), where for any s, s-stage implicit RKN methods of order p = 2s are given. These methods are obtained by writing (1) in first-order form and applying the classical Gauss-Legendre method, taking into account the special form of the differential equation in (1). We shall call the Hairer methods *indirect* Gauss-Legendre methods. Thus, by means of the indirect Gauss-Legendre methods, we are now in the position that we can construct an explicit, parallel RKN method of order 2s requiring s sequential evaluations of **f** per integration step on s processors.

If we are satisfied with parallel RKN methods with a *fixed* number of sequential stages, then we may stop at this stage. However, in actual computation, it

is often more efficient to employ a *dynamic* iteration strategy, that is, we do not set m = [(p-1)/2], but we determine the number of iterations m by the condition that the solution of the corrector method is approximated within a given tolerance. In such an approach, it is desirable to look at the rate of convergence of the iteration process. In our case, the rate of convergence depends on the Butcher matrix (that is, the matrix A appearing in the Butcher array notation of the method). It turns out that the rate of convergence is larger as the value of $\|A^m\|^{1/m}$ is smaller ($\|A^m\|^{1/m}$ will be called the *convergence factor*). To some extent, this can be achieved by using RKN methods whose Butcher matrix A has a small *asymptotic* convergence factor, i.e. methods for which the spectral radius $\rho(A)$ is small. This motivated us to look for corrector methods that possess smaller convergence factors than those of the indirect Gauss-Legendre methods of Hairer. We constructed two families of corrector methods with reduced asymptotic convergence factors.

The first family of methods is obtained by means of *collocation techniques* that are directly applied to the second-order form (1). These methods will be termed *direct* RKN methods. If the collocation points are identified with the Gauss points, then we obtain, like the indirect Gauss-Legendre methods of Hairer, s-stage implicit RKN methods of order p = 2s. However, it turned out that for $p \leq 10$, $\rho(A_{direct})$ is at most 68% of $\rho(A_{indirect})$. In practice, this reduction yields a substantially faster convergence of the fixed point iteration process.

The second family is designed by replacing in an s-stage, implicit RKN method s-k stage values by extrapolation formulas using information from the preceding step. In this way, we obtain a k-stage, implicit, *two-step* RKN corrector. A natural option chooses for the generating RKN method a *direct* or indirect Gauss-Legendre method. Unfortunately, it turns out that the resulting two-step RKN correctors are often zero-unstable. However, by changing the location of the collocation points, we succeeded in finding zero-stable correctors of arbitrarily high order p = 2s having s implicit stages. As for the one-step correctors, the convergence of the fixed point iteration process is controlled by a matrix A occurring in the Butcher-type array notation of the two-step RKN (TRKN) corrector. For $p \leq 10$, we found that $\rho(A_{TRKN})$ is about 30% of $\rho(A_{indirect})$.

Full details of the methods described above, including stability analysis and numerical comparisons, can be found in:

[1] Nguyen huu Cong (1993): *Note on the performance of direct and indirect Runge-Kutta-Nyström methods*, J. Comput. Appl. Math. **45**, 347-355.

[2] Houwen, P.J. van der, Sommeijer, B.P. & Nguyen huu Cong (1991): *Stability of collocation-based Rung Kutta-Nyström methods*, BIT **31**, 469-481.

[3] Nguyen huu Cong (1993): *Explicit parallel two-step Runge-Kutta-Nyström methods*, Report NM-R9401, Centre for Mathematics and Computer Science, Amsterdam, submitted for publication.

2. Parallel implicit Runge-Kutta-Nyström methods

Implicit RKN methods are applied to problems originating from structural mechanics or celestial mechanics, whose solutions possess periodic components with frequencies ranging from small to large, where the lower harmonics are of interest, the higher harmonics are not. Hence, only the solution components corresponding to eigenvalues of the Jacobian matrix $\partial \mathbf{f}/\partial \mathbf{y}$ close to the origin are of interest. In such cases, the ideal method would be a method without dissipation of the lower harmonics (i.e., nonempty periodicity interval), high order of dispersion, and damping of the higher harmonics. The presence of unwanted high harmonics (a form of stiffness) may reduce the step point order considerably. In many stiff problems, it is the stage order that determines the accuracy, rather than the step point order. In order to avoid the effect of order reduction we need methods that have, in addition to a high step point order and the property of unconditional stability, a high stage order. In Sharp-Fine-Burrage (1990), the property of unconditional stability is termed R-stability. In this thesis, we have called it A-stability, in analogy with the terminology used for unconditionally stable methods for first-order initial-value problems.

Following a similar approach as used in Sommeijer (1992) for solving *stiff* first-order problems, we have constructed parallel RKN methods based on iteration of fully implicit RKN methods of collocation type. Such RKN methods possess the largest possible stage order, so that we automatically achieve high stage orders if the RKN method is solved sufficiently accurate. Furthermore, after only a few iterations, the step point order of the iterated method equals that of the underlying RKN corrector. Since there are A-stable RKN methods available of arbitrarily high stage order, the iterated methods satisfy the requirements just mentioned. For an extensive set of suitable RKN methods with high stage orders we refer to reference [2] given above.

The *diagonal-implicit* iteration process that we applied is highly parallel. Firstly, because a large number of the implicit systems that are to be solved in each step, can be solved in parallel, but more important, because all LU-decompositions needed in each step can also be computed in parallel. Hence, only one LU-decomposition per step per processor is required. In fact, after m iterations, the sequential computational complexity of the iterated method consists of m implicit systems of dimension d to be solved in each integration step (d represents the dimension of the differential equation (1)). Thus, the sequential computational complexity is comparable with that of an m-stage diagonally implicit RKN method

whose diagonal entries in the Butcher matrix are all equal. We shall say that the iterated method has m implicit, sequential stages.

There are various strategies for choosing the iteration parameters. One possibility is based on the minimization of the spectral radius of the stage vector iteration matrix. For a large number of indirect RKN correctors taken from the literature, we calculated the iteration parameters with this minimizing property. From these correctors, we selected those which generate methods that remain A-stable after a minimal number of implicit sequential iterations. Let m, p and r denote this minimal number of stages, the step point order, and the stage order, respectively. Then, by means of the minimal-spectral-radius strategy we found methods with (p,r,m) = (3,3,2), (5,3,4) and (5,5,7). By replacing the condition 'the method should *remain* stable after a minimal number of iterations' with the condition 'the method needs only to be A-stable if m is such that the order of the corrector equals that of the iterated method', we found A-stable methods with (p,r,m) = (4,2,2), (6,3,3) and (8,4,4). In order to appreciate these results, we mention the sequential RKN methods (3,1,2) of Crouzeix (1975), (4,1,3) of Sharp-Fine-Burrage (1990), and (5,1,5) and (6,1,5) of Cooper-Sayfy (1979).

Full details of the methods described above can be found in:

[4] Houwen, P.J. van der, Sommeijer, B.P. & Nguyen huu Cong (1992): *Parallel diagonally implicit Runge-Kutta-Nyström methods*, Appl. Numer. Math. **9**, 111-131.

[5] Nguyen huu Cong (1993): *A-stable diagonally implicit Runge-Kutta-Nyström methods for parallel computers*, Numerical Algorithms **4**, 263-281.

3. Future research

During the investigations for this thesis, two further ideas for improving the efficiency of parallel RKN methods arose. Because of the complexity of RKN methods, we decided first to try these ideas out on the less complicated RK methods for first-order differential equations. Moreover, we restricted our investigations to nonstiff problems.

The first idea is to increase the amount of parallelism in step-by-step methods by computing parallel solution values not only at step points, but also at off-step points. Thus, in each step, a whole block of approximations to the exact solution is computed. This approach was successfully used by Enright and Highman (1991) for obtaining reliable defect control in explicit RK methods. Alternatively, this approach can be used for reducing the number of iterations in the iteration process. For example, the block of approximations can be used for obtaining a high-order predictor formula in the next step by some interpolation formulas e.g., Lagrange or Hermite interpolation. By choosing the abscissas of the off-step points narrowly

spaced, we achieve much more accurate predictor values than can be obtained by predictor formulas based on preceding step point values. Moreover, the precise location of the off-step points can be used for minimizing the iteration errors or for maximizing stability boundaries. Since the approximations at the off-step points to be computed in each step can be obtained in parallel, the sequential costs of this block iteration method are equal to those of the iteration methods discussed above. The increased accuracy gives rise to speed-up factors ranging from 2 until 11 when compared with the best sequential methods for first-order problems (cf. reference [6]). However, the number of processors is (of course) much larger.

The second idea again concerns the improvement of the convergence in fixed point iteration of corrector methods. Here, we reduced the magnitude of the asymptotic convergence factor by sacrificing the property of superconvergence of the corrector. Our starting point is an s-stage, symmetric RK (SRK) method based on collocation. For s odd, such SRK methods have order p = s+1 and contain (s-1)/2 free collocation points. These free collocation points can be used for the minimization of the asymptotic convergence factor $\rho(A_{SRK})$. Comparing these minimal asymptotic convergence factors with those associated with fixed point iteration of Gauss-Legendre correctors, we found a reduction to 70% for p = 4 until 50% for p = 10. Again, the price we pay is a larger number of processors to achieve the same order of accuracy (nearly twice as many). However, comparison of numerical results produced by a pth-order, iterated SRK method and a pth-order, iterated Gauss-Legendre method, both implemented on p/2 processors, reveals that the iterated SRK method is superior to the iterated Gauss-Legendre method (see [7]).

The ideas outlined above can be applied to RKN methods and to stiff problems. This will be subject of future research.

[6] Houwen, P.J. van der & Nguyen huu Cong (1993): *Parallel block predictor-corrector methods of Runge-Kutta type*, Appl. Numer. Math. **13**, 109-123.

[7] Nguyen huu Cong (1993): *Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial-value problems*, Report NM-R9320, Centre for Mathematics and Computer Science, Amsterdam, to appear in J. Comput. Appl. Math.

References

Cooper, G.J. & Sayfy, A. (1979): *Semi-explicit A-stable Runge-Kutta methods*, Math. Comp. **33**, 541-556.

Crouzeix, M. (1975): Sur l'approximation des équations différentielles opérationnelles linéaires par des méthodes de Runge-Kutta, Ph.D. Thesis, Université de Paris, France.

Enright, W. H. & Highman, D.J. (1991): *Parallel defect control*, BIT **31**, 647-663.

Hairer, E. (1977): *Méthodes de Nyström pour l'équation différentielle y''=f(x,y)*, Numer. Math. **27**, 283-300.

Hairer, E. (1978): *A Runge-Kutta method of order 10,* J. Inst. Math. Appl. 21, 47-59.

Hairer, E. (1979): *Unconditionally stable methods for second order differential equations*, Numer. Math. 32, 373-379.

Hairer, E. (1982): *A one-step method of order 10 for y'' = f(x,y)*, IMA J. Numer. Anal. **2**, 83-94.

Sharp, P. W., Fine, J. M. & Burrage, K. (1990): *Two-stage and three-stage diagonally implicit Runge-Kutta-Nyström methods of orders three and four*, IMA J. Numer. Anal. **10**, 489-504.

Sommeijer, B. P. (1992): *Parallelism in the Numerical Integration of Initial Value Problems*, Doctoral thesis, University of Amsterdam.

CHAPTER I

Stability of collocation-based Runge-Kutta-Nyström methods

published in: *BIT* **31** (1991), 469-481

STABILITY OF COLLOCATION-BASED RUNGE-KUTTA-NYSTRÖM METHODS

P. J. VAN DER HOUWEN[1], B. P. SOMMEIJER[1] and NGUYEN HUU CONG*[2]

[1] *Centre for Mathematics and Computer Science, Box 4079, 1009 AB Amsterdam, The Netherlands*

[2] *Faculty of Mathematics, Mechanics and Informatics University of Hanoi, Thuong dinh, Dong Da, Hanoi, Vietnam*

Abstract.

We analyse the attainable order and the stability of Runge-Kutta-Nyström (RKN) methods for special second-order initial-value problems derived by collocation techniques. Like collocation methods for first-order equations the step point order of s-stage methods can be raised to $2s$ for all s. The attainable stage order is one higher and equals $s + 1$. However, the stability results derived in this paper show that we have to pay a high price for the increased stage order.

AMS Subject classification: 65M10, 65M20.

1. Introduction.

In this paper we shall be concerned with the analysis of implicit Runge-Kutta-Nyström (RKN methods) based on collocation for integrating the initial-value problem (IVP) for systems of special second-order, ordinary differential equations (ODEs) of dimension d, i.e. the problem,

$$\boldsymbol{y}''(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{y}'(t_0) = \boldsymbol{u}_0, \tag{1.1}$$

$$\boldsymbol{y}: \mathbb{R} \to \mathbb{R}^d, \quad \boldsymbol{f}: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d, \quad t_0 \le t \le T.$$

Our motivation for studying implicit RKN methods is the arrival of parallel computers which enables us to solve the implicit relations occurring in the stage vector equation quite efficiently, so that, what is so far considered as the main disadvantage of fully implicit RKN methods, is reduced a great deal. We consider two types of collocation methods for second-order equations: methods based on *direct* collocation and on *indirect* collocation (that is, methods obtained by writing the special second-order equation in first-order form and by applying collocation methods for first-order equations [6]). The theory of *indirect* collocation methods

*) These investigations were supported by the University of Amsterdam who provided the third author with a research grant for spending a total of two years in Amsterdam.

Received September 1990. Revised February 1991.

for problem (1.1) completely parallels the well-known theory of collocation methods for first-order equations (cf. [3], [7]). The attainable step point and stage order using s stages equals $2s$ and s. For all s, these methods can be made A-stable and of order $2s$ (Gauss-type methods) or L-stable and of order $2s - 1$ (Radau IIA type methods) by a suitable choice of the collocation parameters. There even exist indirect collocation methods with stage order s using only $s - 1$ implicit stages (and one explicit stage) which are known to be A-stable for $s \leq 9$ (Newton-Cotes methods [15]) or strongly A-stable for $s \leq 5$ (Lagrange methods [9]). In the following, k will denote the number of implicit stages of the method. Since in actual computation, it is the number of *implicit* stages that determines the computational complexity of the method, we shall often characterize RKN methods by k rather than by s.

The stability of direct collocation was investigated in Kramarz [12] (see also [1]). The main object of the present paper is to extend the work of Kramarz and to derive order and stability results for direct collocation methods. It will be shown that the attainable step point order is similar to that of indirect collocation methods, but the stage order can be raised to $s + 1$ leaving all but one collocation parameters free. High stage orders are attractive in the case of stiff problems, provided that the method is A or P-stable. However, it seems that the increased-stage-order methods all have *finite* stability boundaries. If the stage order is decreased to s, then infinite stability boundaries can be obtained. We found A-stable methods with $k = s = 2$, $k = s = 3$ and with $k = s - 1 = 4$ implicit stages.

We also investigated two stabilizing techniques for achieving A-stability. The first stabilizing technique is based on the *preconditioning* of the right-hand side in (1.1), that is, stiff components in the right-hand side are damped. In this way, it is possible to transform conditionally stable RKN methods into unconditionally stable preconditioned RKN methods (*PRKN methods*) at the cost of a slightly more complicated relation for the stage vector. The second stabilizing technique is based on the combination of different, conditionally stable RKN methods. We will give examples of A-stable, composite methods (*CRKN methods*) with stage order s and $k = s - 1$ implicit stages for $k \leq 4$.

Summarizing, this paper investigates three families of methods based on direct collocation. Assuming that they all use k implicit stages (including those the CRKN methods are composed of), we get the following survey of main characteristics (p and r denote the step point and stage orders):

Table 1.1. Survey of characteristics of methods based on direct collocation

Family		s	p	r	Stability	With preconditioning	Subsections
A. single:	Gauss	k	$2k$	$k+1$	Conditionally stable	Weakly A-stable	4.2.1, 4.3
	Radau	k	$2k-1$	$k+1$	Conditionally stable	Weakly A-stable	4.2.1, 4.3
	Lobatto	$k+1$	$2k$	$k+2$	Conditionally stable	Weakly A-stable	4.2.1, 4.3
B. single:	$k = 2,3$	k	k	k	Strongly A-stable	–	4.2.2
	$k = 4$	$k+1$	$k+1$	$k+1$	Strongly A-stable	–	4.2.2
C. composite:	$k \leq 4$		$k+1$	$k+1$	Strongly A-stable	–	4.2.3

2. RKN methods.

For the sake of simplicity of notation, we assume that (1.1) is a scalar problem. However, all considerations can be trivially extended to systems of equations. For scalar ODEs, the general s-stage RKN method is defined by

$$\begin{aligned} &y_{n+1} = y_n + hy'_n + h^2 \boldsymbol{b}^T f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}), \quad y'_{n+1} = y'_n + h\boldsymbol{d}^T f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}), \\ &\boldsymbol{Y} = \boldsymbol{e}y_n + \boldsymbol{c}hy'_n + h^2 A f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}), \end{aligned} \tag{2.1}$$

where h is the stepsize, $\{t_n\}$ is the set of step points and y_{n+1}, y'_{n+1} denote the numerical approximations to $y(t_{n+1})$, $y'(t_{n+1})$. Furthermore, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ are s-dimensional vectors, $\boldsymbol{e}$ is the s-dimensional vector with unit entries, A is an $s \times s$ matrix, and, for any pair of vectors $\boldsymbol{v} = (v_i)$, $\boldsymbol{w} = (w_i)$, $f(\boldsymbol{v}, \boldsymbol{w})$ denotes the vector with entries $f(v_i, w_i)$.

If the last row of A equals the row vector $\boldsymbol{b}^T$, i.e., $\boldsymbol{b}^T = \boldsymbol{e}_s^T A$, then, as in the case of RK methods for first-order IVPs, such methods are said to be *stiffly accurate*. In general, stiffly accurate methods perform better on stiff problems than methods that are not stiffly accurate.

2.1. Order of accuracy.

Let $\boldsymbol{Y}(t_{n+1})$ denote the vector with components $y(t_n + c_i h)$ with y the locally exact solution of (1.1) satisfying $y(t_n) = y_n$ and $y'(t_n) = y'_n$, and suppose that the local errors are given by

$$\begin{aligned} &y(t_{n+1}) - y_{n+1} = O(h^{p_1+1}), \quad y'(t_{n+1}) - y'_{n+1} = O(h^{p_2+1}), \\ &\boldsymbol{Y}(t_{n+1}) - \boldsymbol{e}y_n - \boldsymbol{c}hy'_n - h^2 A f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}(t_{n+1})) = O(h^{p_3+1}), \end{aligned} \tag{2.2}$$

then the (global) order of accuracy p and the (global) stage order r are respectively defined by $p = \min\{p_1, p_2\}$ and $r = \min\{p_1, p_2, p_3\}$. Notice that the *local* stage order equals $p_3 + 1$.

For stiff *first-order* ODEs the accuracy reducing effect of order reduction for methods with low stage orders is well known [4], and therefore collocation methods with their high stage orders are rather accurate for stiff problems. A similar phenomenon occurs in stiff *second-order* equations (cf. Example 2.1 in [10]).

2.2. Linear stability.

The linear stability of RKN methods is investigated by applying them to the test equation $y'' = \lambda y$, where λ runs through the eigenvalues of $\partial \boldsymbol{f}/\partial \boldsymbol{y}$. This leads to a recursion of the form

$$\begin{aligned} &\boldsymbol{v}_{n+1} = M(z)\boldsymbol{v}_n, \quad \boldsymbol{v}_n := (y_n, hy'_n)^T, \\ &M(z) := \begin{pmatrix} 1 + z\boldsymbol{b}^T(I - Az)^{-1}\boldsymbol{e} & 1 + z\boldsymbol{b}^T(I - Az)^{-1}\boldsymbol{c} \\ z\boldsymbol{d}^T(I - Az)^{-1}\boldsymbol{e} & 1 + z\boldsymbol{d}^T(I - Az)^{-1}\boldsymbol{c} \end{pmatrix}, \end{aligned} \tag{2.3}$$

where $z := \lambda h^2$. The damping effect of the matrix $M(z)$ can be characterized by the *stability function* $R(z)$ of the RKN method defined by the spectral radius $\rho(M(z))$ of $M(z)$.

DEFINITION 2.1. The collection of points on the negative real z-axis is called
(i) the region of *stability* if in this region $R(z) := \rho(M(z)) < 1$,
(ii) the region of *periodicity* if $R(z) = 1$ and $[\text{trace } M(z)]^2 - 4 \det M(z) < 0$.

If $(-\beta_{\text{stab}}, 0)$ lies in the stability region, then β_{stab} is called the *stability boundary*, and if $(-\beta_{\text{per}}, 0)$ lies in the periodicity region, then β_{per} is called the *periodicity boundary*. *If* $\beta_{\text{stab}} = \infty$, then the RKN method is called *A-stable* and if $\beta_{\text{per}} = \infty$, then it is called *P*-stable. An *A*-stable RKN method is called *L-stable* if $R(-\infty) = 0$.

3. RKN methods based on collocation.

3.1. Indirect collocation methods.

Indirect collocation methods are generated by applying an RK collocation method to the first-order representation of (1.1). Thus, writing (1.1) in the form

$$(1.1') \qquad \boldsymbol{y}'(t) = \boldsymbol{u}(t), \qquad \boldsymbol{u}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \qquad \boldsymbol{u}(t_0) = \boldsymbol{u}_0,$$

and applying an RK method for first-order equations:

$$y_{n+1} = y_n + h\boldsymbol{d}^T f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}), \qquad \boldsymbol{Y} = \boldsymbol{e}y_n + h\mathring{A} f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}),$$

we obtain an RKN method of the form (2.1) with (cf. [6])

$$(3.1) \qquad \boldsymbol{b} = \mathring{A}^T \boldsymbol{d}, \qquad A = \mathring{A}^2.$$

Notice that when the generating RK method has order p and k implicit stages, then this is true for the RKN method as well. Now, let the generating RK method be a collocation method based on the s distinct collocation points $\{t_{nj} := t_n + c_j h, j = 1, \ldots, s\}$, then (see e.g. [7])

$$(3.2) \qquad \mathring{A} = (\mathring{a}_{ij}) := (\alpha_j(c_i)), \qquad \boldsymbol{d} = (d_i) := (\alpha_i(1)), \qquad \alpha_j(x) := \int_0^x L_j(\xi)\, d\xi,$$

$$L_j(x) := \prod_{i=1, i\neq j}^{s} \frac{x - c_i}{c_j - c_i},$$

where $i, j = 1, \ldots, s$. The family of indirect collocation methods defined by (3.1) and (3.2) has order $p = r = s$ for all collocation vectors $\boldsymbol{c}$ (see e.g. [4]). The RKN method will be called *symmetric* if the location of the collocation points t_{nj} is symmetric with respect to $t_n + h/2$.

By a special choice of the collocation points, it is possible to increase the step point order p beyond s (superconvergence at the step points). The following theorem holds (see e.g. [7, p. 207]):

THEOREM 3.1. *The indirect RKN method defined by* $\{(3.1),(3.2)\}$ *has global step point order and global stage order* $p = r = s$ *for all sets of distinct collocation parameters* c_i. *We have* $p = s + q$ *if, in addition,*

$$P_j(1) = 0, \qquad P_j(x) := \int_0^x \xi^{j-1} \prod_{i=1}^{s} (\xi - c_i)\, d\xi, \qquad j = 1, 2, \ldots, q. \tag{3.3}$$

3.2. Direct collocation methods.

3.2.1. Methods of order $p = r = s$. Following [2, p. 241], let S be the space of real, piecewise continuously differentiable polynomials of degree not exceeding $s + 1$ associated with the set of intervals $[t_n, t_{n+1}]$. Thus, if u is in S, then $u(t)$ is a polynomial of degree $\leq s + 1$ on each interval $[t_n, t_{n+1}]$, $n = 0, \ldots, N - 1$. For such functions u, the second derivative u'' is a polynomial of degree not exceeding $s - 1$ on each of the intervals $[t_n, t_{n+1}]$. Using the $L_j(x)$ defined in (3.2) we may write

$$\begin{aligned} u''(t_n + xh) &= \sum_{j=1}^{s} L_j(x) u''(t_{nj}), \qquad u'(t_n + xh) = u'(t_n) + h \sum_{j=1}^{s} \alpha_j(x) u''(t_{nj}), \\ u(t_n + xh) &= u(t_n) + xhu'(t_n) + h^2 \sum_{j=1}^{s} \beta_j(x) u''(t_{nj}), \end{aligned} \tag{3.4}$$

where $\alpha_j(x)$ is defined in (3.2) and

$$\begin{aligned} \beta_j(x) &:= \int_0^x \int_0^\eta L_j(\xi)\, d\xi\, d\eta = \int_0^x \int_\xi^x L_j(\xi)\, d\eta\, d\xi = \int_0^x (x - \xi) L_j(\xi)\, d\xi \\ &= x\alpha_j(x) - \int_0^x \xi L_j(\xi)\, d\xi. \end{aligned} \tag{3.5}$$

Next, we require that the function u satisfies the collocation equations $u''(t_{nj}) = f(t_{nj}, u(t_{nj}))$ for $j = 1, \ldots, s$. Then (3.4) leads to:

$$\begin{aligned} u(t_{ni}) &= u(t_n) + c_i h u'(t_n) + h^2 \sum_{j=1}^{s} \beta_j(c_i) f(t_{nj}, u(t_{nj})), \\ u'(t_{ni}) &= u'(t_n) + h \sum_{j=1}^{s} \alpha_j(c_i) f(t_{nj}, u(t_{nj})), \qquad i = 1, \ldots, s, \\ u(t_{n+1}) &= u(t_n) + hu'(t_n) + h^2 \sum_{j=1}^{s} \beta_j(1) f(t_{nj}, u(t_{nj})), \\ u'(t_{n+1}) &= u'(t_n) + h \sum_{j=1}^{s} \alpha_j(1) f(t_{nj}, u(t_{nj})). \end{aligned} \tag{3.6}$$

By writing $y_n := u(t_n)$, $y_n' := u'(t_n)$ and $\boldsymbol{Y} := (u(t_{ni}))$ and by introducing the quantities

$$\boldsymbol{b} := (b_i), \quad \boldsymbol{d} := (d_i), \quad A = (a_{ij}), \quad b_i := \beta_i(1), \quad d_i := \alpha_i(1), \quad a_{ij} := \beta_j(c_i), \tag{3.7}$$

the method (3.6) is recognized as the s-stage RKN method (2.1). As in the case of indirect collocation methods, the RKN method defined by (3.7) will be called *symmetric* if the location of the collocation points t_{nj} is symmetric with respect to $t_n + h/2$.

Since in the interval $[t_n, t_{n+1}]$ the function u is a polynomial of degree $\leq s+1$ satisfying the collocation equations, it follows that $p_1 = s+1$, $p_2 = s$ and $p_3 = s+1$. Hence, locally, the order of the y'-component is one lower than the order of the other components. Therefore, we have the global order result $p = r = s$ (see also Subsection 2.1).

THEOREM 3.2. *The direct RKN method defined by* (3.7) *has global step point order and global stage order $p = r = s$ for all sets of distinct collocation parameters c_i.*

3.2.2. Superconvergence. As in the case of indirect collocation, it is possible to increase the orders p_1 and p_2 beyond $s+1$ and s by a special choice of the collocation points (superconvergence at the step points). We first consider the local order of y'_{n+1} by writing the local error of y'_{n+1} in the form

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\,dt = \dot{y}(t_{n+1}) - y'_n = h\boldsymbol{d}^T f(\boldsymbol{e}t_n + \boldsymbol{c}h, \boldsymbol{Y}) + O(h^{p_2+1}). \tag{3.8}$$

It can be shown that $\boldsymbol{d}$ generates a quadrature formula with quadrature error of $O(h^{s+q+1})$ whenever the collocation points satisfy the relations (3.3), i.e., $p_2 = s+q$. Thus, setting $q = 1$, we have:

THEOREM 3.3. *If* (3.3) *is satisfied for $q = 1$, then the direct RKN method defined by* (3.7) *has global step point order and global stage order $p = r = s+1$. For all symmetric methods with an odd number of stages, condition* (3.3) *is satisfied for $q = 1$.*

EXAMPLE 3.1. For $s = 2$ and $q = 1$ condition (3.3) yields $c_2 = (2 - 3c_1)/(3 - 6c_1)$. Choosing $c_1 = 0$, we find that $c_2 = 2/3$. Thus, the direct collocation method with $\boldsymbol{c} = (0, 2/3)^T$ has order $p = r = 3$ and requires only one implicit stage. Furthermore, for $\boldsymbol{c} = (1/3, 1)^T$ a stiffly accurate method results with order $p = r = 3$.

THEOREM 3.4. *If condition* (3.3) *is satisfied, then the direct RKN method* (3.7) *has global step point order $p = s+q$.*

PROOF. From (3.3) it follows that $p_2 = s+q$ (cf. (3.8)). Furthermore, the condition $P_1(1) = 0$ implies

$$\int_0^1 \prod_{i=1}^{s} (\xi - c_i)\,d\xi = \int_0^1 (\xi - c_j) \prod_{i=1, i\neq j}^{s} (\xi - c_i)\,d\xi = 0.$$

Hence, from the definition of the Lagrange polynomials L_j in (3.2) it follows that

$$\int_0^1 \xi L_j(\xi)\,d\xi - \int_0^1 c_j L_j(\xi)\,d\xi = 0. \tag{3.9}$$

By observing that (cf. (3.7))

$$b_i = \beta_i(1) = \alpha_i(1) - \int_0^1 \xi L_i(\xi)\,d\xi, \qquad d_i = \alpha_i(1) = \int_0^1 L_i(\xi)\,d\xi,$$

we derive from (3.9) that $b_i = d_i - d_i c_i$ for $i = 1, \ldots, s$. This condition is recognized as a well-known simplifying condition for RKN methods (see, e.g. [7, p. 268]). According to a lemma of Hairer [5], this simplifying condition implies that the order conditions for the y-component are a subset of the order conditions for the y'-component. Thus, if $p_2 = s + q$, then $p_1 = s + q$, so that the assertion of the theorem is proved. ■

COROLLARY 3.1. *Direct and indirect collocation methods with the same collocation points have the same step point order. The stage order of direct collocation methods is one higher whenever* $P_1(1) = 0$.

For a numerical example illustrating this corollary, we refer to [10].

4. Stability of collocation methods.

4.1. Indirect collocation.

In the case of the indirect collocation methods, we can resort to the theory of collocation methods for first-order equations and the derivation of suitable methods is straightforward. For indirect methods of the form (3.1) it can be derived that the matrix $M(z)$ defined in (2.3) is given by

$$M(z) = R^*(Z), \quad Z := \begin{pmatrix} 0 & 1 \\ z & 0 \end{pmatrix}, \quad R^*(w) := 1 + w\boldsymbol{b}^T(I - Aw)^{-1}\boldsymbol{e}, \quad z := \lambda h^2, \tag{4.1}$$

where $R^*(w)$ denotes the stability function of the generating RK method. Hence, the stability function of the generated RKN method is given by $R(z) := \rho(M(z)) = \operatorname{Max}\{R^*(\pm\sqrt{z})\}$. From this formula, we conclude that if, and only if, (2.1) possesses the stability interval $(-\beta_{\text{stab}}, 0)$, then the generating RK method possesses the imaginary stability boundary $(\beta_{\text{stab}})^{1/2}$. Hence, A-stable RK methods (i.e., $(\beta_{\text{stab}})^{1/2} = \infty$) generate A-stable RKN methods. In particular, the s-stage Radau IIA methods generate L-stable RKN methods with step point order $2s - 1$ and stage order equal to the number of implicit stages s. However, the Lagrange methods

derived in [9] generate (strongly) A-stable RKN methods where the stage order equals the number of implicit stages plus one. If one wants RKN methods with a nonempty periodicity interval, we have to choose generating RK methods with stability functions that have modulus 1 along the imaginary axis. This means that $R^*(w)$ should satisfy the (necessary and sufficient) condition $R^*(w)R^*(-w) = 1$, that is, the collocation points should be distributed symmetrically with respect to 1/2 (see also [16], where an analytical expression for $R^*(w)$ is derived, merely in terms of the collocation points). For example, the diagonal elements of the Padé table associated with $\exp(w)$ satisfy this condition, and hence, the s-stage Gauss-Legendre methods generate s-stage, P-stable RKN methods with stage order s and step point order $2s$ (cf. [6]).

4.2. Direct collocation.

Similar to the analysis performed by Wright [16] in the case of first order ODEs, it is possible to derive closed form expressions for the RKN parameters in terms of the collocation vector $\boldsymbol{c}$ (see the Appendix to [10], where full details can be found). With the help of these expressions, the matrix M and its spectral radius can, at least formally, be expressed in terms of $\boldsymbol{c}$. However, the complexity of these expressions is beyond a manageable level. Therefore, we resorted to numerical search techniques. Especially in the derivation of methods with three or more stages, we think this is the only practical approach. As a result of this numerical search, it turned out that the situation for direct collocation methods is less favourable than for indirect methods; the construction of direct collocation methods which are A-stable or P-stable and have RKN parameters of acceptable magnitude (say, not greater than 10 in magnitude) is quite cumbersome. For instance, we did not find stiffly accurate methods in the family A of Table 1.1 that are A-stable or P-stable. For two-stage methods this is immediate from a result of Kramarz [12], who proved that two-stage, stiffly accurate methods (i.e., $c_2 = 1$) can only be A-stable if $0.7 \leq c_1 < 1$. This conflicts with the requirement to obtain $p = r = 3$ which needs $c_1 = 1/3$ (see Example 3.1).

4.2.1. Conditionally stable RKN methods. In Table 4.1 order and stability characteristics of methods generated by conventional sets of collocation points are listed (these methods belong to family A of Table 1.1). In general, these methods have a number of intervals of instability of which the first two are listed. They are indicated by U_1 and U_2, and the corresponding maximum values of the stability function R are denoted by $R_{\max}(U_i)$. These stability results indicate that, from a practical point of view, direct collocation methods based on Gauss, Radau and Lobatto collocation points are of limited value, because the rather small stability or periodicity boundaries make them unsuitable for stiff problems (which is the main class of problems where implicit RKN methods are used). The A-stable, indirect analogues are clearly more suitable for integrating stiff problems. However, in Section 4.3, we shall describe a stabilizing technique based on preconditioning matrices that removes stiff components from the right-hand side function and

transforms conditionally stable methods into A-stable or P-stable methods. By means of this technique the methods from Table 4.1 can be made A-stable or P-stable.

Table 4.1. Order and stability characteristics of direct Gauss, Radau and Lobatto collocation methods.

	Method	c^T	p	r	U_1	$R_{\max}(U_1)$	U_2	$R_{\max}(U_2)$	$R(\infty)$
$k=2$	Gauss	cf. [4]	4	3	$(-12,-9)$	1.23	$(-\infty,-35.9)$	13.9	13.9
	Radau	cf. [4]	3	3	$(-16.73,-8.61)$	1.25	$(-\infty,-108)$	2.0	2.0
	Lobatto	cf. [4]	4	4	$(-12.0,-9.6)$	1.17	$(-\infty,-48)$	7.9	7.9
$k=3$	Gauss	cf. [4]	6	4	$(-10.01,-9.77)$	1.01	$(-60.1,-34.2)$	2.1	26.0
	Radau	cf. [4]	5	4	$(-10.32,-9.55)$	1.04	$(-103.1,-34.9)$	1.97	3.0
	Lobatto	cf. [4]	6	5	$(-10,-9.82)$	1.01	$(-\infty,-37.5)$	13.9	13.9
$k=4$	Gauss	cf. [7]	8	5	$(-9.876,-9.865)$	1.0007	$(-42.1,-37.8)$	1.17	42.0
	Radau	cf. [9]	7	5	$(-9.90,-9.84)$	1.002	$(-45.8,-36.5)$	1.29	4.0
	Lobatto	cf. [9]	8	6	$(-9.876,-9.866)$	1.0006	$(-42,-38.5)$	1.13	21.9

4.2.2. A-stable RKN methods with $p=r=s$. If we drop the additional order condition (3.3), then the orders are given by $p_1 = p_3 = s+1$ and $p_2 = s$ (see Section 2.1), so that $p = r = s$ (family B of Table 1.1). We found A-stable methods with $k = s$ implicit stages for $k = 2$ and $k = 3$, and an A-stable method with $k = s-1$ implicit stages for $k = 4$. These are respectively generated by $c^T = (3/4, 1)$, $c^T = (-1/5, 9/10, 1)$ and $c^T = (-1/4, 0, 9/10, 19/20, 1)$ (for more details we refer to the Appendix to [10]). In the following subsection these methods are compared with methods based on composition of RKN methods.

4.2.3 A-stable composite methods with $p=r=k+1$. It is sometimes possible to construct methods with improved stability properties by composing a new method from a sequence of given RKN methods (preferably with equal numbers of implicit stages). In order to define these *composite* RKN methods (CRKN methods), we write the RKN method (2.1) in the compact form $\boldsymbol{w}_{n+1} = L(h, \boldsymbol{w}_n)$, $\boldsymbol{w}_n := (y_n, y'_n)^T$, where L is a (nonlinear) operator defined by the RKN method. Suppose that we are given v RKN methods (not necessarily with the same number of stages) characterized by operators L_i and all of order p. Then we may define the methods $\boldsymbol{w}_{n+i} = L_i(h, \boldsymbol{w}_{n+i-1})$ for $n = 0, v, 2v, \ldots$, and $i = 1, \ldots, v$. Evidently, these CRKN methods are again of order p. Applying the CRKN method to the equation $y'' = \lambda y$, we may write $\boldsymbol{w}_{n+i} = M_i(z)\boldsymbol{w}_n$ where as before $z := \lambda h^2$ and where the $M_i(z)$ denote the amplification matrices of the individual methods. The stability function becomes the spectral radius of the product of the matrices $M_i(z)$ with $i = v, v-1, \ldots, 1$. Presenting CRKN methods by the formula $\prod c_i^T$, where the c_i correspond to the individual RKN methods, we found three suitable A-stable CRKN methods with $p = r = k+1$ (family C of Table 1.1). These are generated by: $(1/3,\ 1) * (0,\ 19/20,\ 1)^2$,

$(0, 1/2, 19/20, 1)*(0, 9/10, 19/20, 1)^2$ and $(1/10, 26/53, 19/20, 1)*(0, 1/4, 9/10, 19/20, 1)^2$. The first two methods improve on the $k = 2$ and $k = 3$ methods of family B. We remark that the collocation vector $(1/10, 26/53, 19/20, 1)$ occurring in the third method satisfies condition (3.3) for $q = 1$ (for more details we refer to the Appendix to [10]).

EXAMPLE 4.1. The A-stable methods of the families B and C are applied to the semidiscretization of

$$\frac{\partial^2 u}{\partial t^2} = \frac{u^2}{1 + 2x - 2x^2}\frac{\partial^2 u}{\partial x^2} + u(4\cos^2(t) - 1), \quad 0 \le t \le 2\pi, \quad 0 \le x \le 1, \tag{4.2}$$

with initial and Dirichlet boundary conditions such that the solution is given by $u = (1 + 2x - 2x^2)\cos(t)$. Using 3-point symmetric spatial discretization on grid points $x_j = j/20$, we obtain a set of 19 ODEs.

Table 4.2. NCD values produced by A-stable methods from the families B and C for Problem (4.2).

	Method	p	r	$h = \pi/15$	$h = \pi/30$	$h = \pi/60$
$k = 2$	$(3/4, 1)$	2	2	*	3.6	4.1
	$(1/3, 1)*(0, 19/20, 1)^2$	3	3	3.7	4.6	5.5
$k = 3$	$(-1/5, 9/10, 1)$	3	3	*	4.4	5.3
	$(0, 1/2, 19/20, 1)*(0, 9/10, 19/20, 1)^2$	4	4	6.3	7.3	8.5
$k = 4$	$(-1/4, 0, 9/10, 19/20, 1)$	5	5	6.9	8.4	9.9
	$(1/10, 26/53, 19/20, 1)*(0, 1/4, 9/10, 19/20, 1)^2$	5	5	7.8	9.2	10.8

Table 4.2 lists the number of correct digits (NCD) obtained at the end of the integration interval, i.e., the value defined by $\mathrm{NCD}(h) := -\log_{10}(\|\text{ global error (obtained with stepsize } h) \text{ at } t = t_{\mathrm{end}}\|_\infty)$. An asterisk denotes an unstable behaviour.

The composite methods perform rather well, in particular in the cases $k = 2$ and $k = 3$.

4.3. *A*-stable preconditioned methods.

As observed above, RKN methods based on direct collocation methods often have finite stability boundaries. A simple technique for constructing methods with large stability boundaries replaces the scalar parameters in an RKN method by matrix operators, usually functions of h and of the Jacobian matrix of the system of ODEs. In [8] such methods were called *generalized RK(N) methods*. Special cases are the celebrated Rosenbrock methods [14] and the Liniger-Willoughby methods [13]. In this paper, we consider generalized RKN methods obtained by replacing in the RKN method all righthand side evaluations $\boldsymbol{f}$ by $S\boldsymbol{f}$ (see also [11] where related right-hand side smoothings are discussed). The preconditioning matrix S is required to be such that $S\boldsymbol{f}$ converges to $\boldsymbol{f}$ as h tends to 0. Furthermore, to be effective,

S should strongly damp the "high frequency (or, stiff) components" (i.e., eigenvectors of the Jacobian corresponding to eigenvalues of large modulus). On the other hand, to preserve accuracy, S should have a negligible effect on the "low frequency components" (eigenvectors corresponding to eigenvalues of small modulus). This leads us to a preconditioning matrix of the form

$$S = [T(h^2 J_n)]^{-1}, \qquad T(z) := 1 + \varepsilon(-z)^\sigma, \qquad J_n := \frac{\partial \boldsymbol{f}(t_n, \boldsymbol{y}_n)}{\partial \boldsymbol{y}}, \tag{4.3}$$

where ε is a small (nonnegative) number, σ is a positive integer, and the minus sign in front of z is added to make T nonsingular for all negative z. The resulting method will be called a *preconditioned* RKN method (PRKN method). The following theorem presents a condition for A- and P-stability.

THEOREM 4.1. *Given an RKN method with step point and stage order p, with stability boundary β_{stab}, and with periodicity boundary β_{per}. The PRKN method generated by* (4.3) *has step point and stage order p if $2\sigma \geq p$, and it is A-stable if ε is bounded below by $(\sigma - 1)^{\sigma-1}(\sigma\beta_{\text{stab}})^{-\sigma}$. The method is P-stable if in this lower bound β_{stab} is replaced by β_{per}, provided that $\beta_{\text{per}} \neq 0$.*

PROOF. Evidently, by replacing $\boldsymbol{f}$ by $S\boldsymbol{f}$, we introduce local perturbations at worst of $O(h^{p+1})$, so that the global step point and stage order of the PRKN method is still p. Furthermore, if the PRKN method is applied to the test equation $y'' = \lambda y$, then the recursion (2.3) assumes the form

$$\boldsymbol{v}_{n+1} = M(\zeta(z))\boldsymbol{v}_n, \qquad \boldsymbol{v}_n := (y_n, hy'_n)^T, \qquad z := \lambda h^2, \qquad \zeta(z) := \frac{z}{1 + \varepsilon(-z)^\sigma}.$$

The corresponding stability function takes the form $R^*(z) := \rho(M(\zeta(z))) = R(\zeta(z))$, where $R(z)$ denotes the stability function of the original RKN method. The stabilized RKN method is A-stable if $\zeta(z)$ satisfies the inequality $-\beta_{\text{stab}} \leq \zeta(z) \leq 0$, where β_{stab} denotes the stability boundary of the original RKN method. It is easily verified that this leads to the lower bound for ε of the theorem. By replacing β_{stab} by β_{per}, and by observing that the values of R^* on the negative z-axis are composed of the values of R on the interval $(-\beta_{\text{per}}, 0)$ which equal 1, it is immediate that we have P-stability. ■

EXAMPLE 4.2. In order to see the effect of the preconditioning technique on the accuracy we choose a conditionally stable method from family A (see Table 1.1), and we perform computations with and without preconditioning. The sequence of stepsizes is chosen such that for certain values of h (in the table of results indicated in bold face) the eigenvalues of $h^2 J_n$ enter the region of instability U of the method. By choosing large integration intervals, we achieve that there are sufficiently many steps to develop instabilities when the region U is entered. Hence, we expect a sudden drop of accuracy when this happens. If preconditioning is applied, then such a drop of accuracy should not occur. Table 4.3 lists results for the problem [12]

$$(4.4)\quad y''(t) = \begin{pmatrix} 2498 & 4998 \\ -2499 & -4999 \end{pmatrix} y'(t), \quad y(0) = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \quad y'(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad 0 \le t \le 100,$$

with exact solution $y(t) = (2\cos(t), -\cos(t))^T$. Without preconditioning, the direct 3-stage Radau method is unstable for the stepsize $h = 1/6$ and $h = 1/15.8$, that is at the points $z = -69.4$ and $z = -10$ (cf. Table 4.1). These results show that A-stability is retained by preconditioning without reduction of the accuracy. We also applied the indirect version of the 3-stage Radau method (which is L-stable and does not need preconditioning). It turned out to perform slightly less accurate than its preconditioned, direct counterpart.

Table 4.3. NCD values produced by the 3-stage (A) and indirect (B) Radau methods for Problem (4.4).

Method	ε	σ	$h^{-1} =$	4	**6**	11	15.4	**15.8**	16.2	20
			$-z =$	156	**69.4**	20.7	10.5	**10**	9.5	6.25
A	0			5.2	*	7.4	8.2	*	8.3	8.7
A	0.0002	3		5.1	**6.0**	7.4	8.1	**8.2**	8.2	8.7
A	0.000015	4		5.2	**6.1**	7.4	8.2	**8.2**	8.3	8.7
B	–	–		4.6	5.5	6.8	7.6	7.6	7.7	8.1

In addition to the autonomous problem (4.4), we also performed a test with a *nonautonomous* variant of this problem. For that purpose, we added the term $-\gamma(y_1 - 2\cos(t), y_2 + \cos(t))^T$ to the right-hand side of (4.4). Notice that this does not change the exact solution. For γ-values up to, say, 100, the preconditioned methods show a similar accuracy as for the autonomous problem, but quickly loose accuracy if γ increases. The reason is, of course, that for such large γ-values the right-hand side is dominated by the nonautonomous term, whereas its influence does not enter into the preconditioning matrix S. The indirect method, on the other hand, performs very well, also for large γ-values (full details on this experiment can be found in [10]).

Summarizing, we conclude that the preconditioning technique is a useful tool (i.e., for retaining A-stability without loosing accuracy) for problems where the Jacobian matrix is constant or slowly varying (with respect to the stepsize) and where the nonautonomous (inhomogeneous) term is also of moderate variation.

REFERENCES

[1] Boor, C. R. de & Swartz, B. (1973): *Collocation at Gaussian points*, SIAM J. Numer. Anal. 10, 582–606.

[2] Brunner, H & Houwen, P. J. van der (1986): *The Numerical Solution of Volterra Equations*, North-Holland, Amsterdam.

[3] Butcher, J. C. (1987): *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods*, Wiley, New York.

[4] Dekker, K. & Verwer, J. G. (1984): *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam.
[5] Hairer, E. (1977): *Méthodes de Nyström pour l'équation différentielle* $y'' = f(x, y)$, Numer. Math. 27, 283–300.
[6] Hairer, E. (1979): *Unconditionally stable methods for second order differential equations*, Numer. Math. 32, 373–379.
[7] Hairer, E., Nørsett, S. P. & Wanner, G. (1987): *Solving Ordinary Differential Equations*, I. *Nonstiff Problems*, Springer-Verlag, Berlin.
[8] Houwen, P. J. van der (1977): *Construction of Integration Formulas for Initial Value Problems*, North-Holland, Amsterdam.
[9] Houwen, P. J. van der & Sommeijer, B. P. (1991): *Iterated Runge-Kutta methods on parallel computers*, to appear in SSISC.
[10] Houwen, P. J. van der, Sommeijer, B. P. & Nguyen huu Cong (1990): *Stability of collocation-based Runge-Kutta-Nyström methods*, Report NM-R9016, Centre for Mathematics and Computer Science, Amsterdam.
[11] Jameson, A. (1983): *The evolution of computational methods in aerodynamics*, J. Appl. Mech. 50, 1052–1076.
[12] Kramarz, L. (1980): *Stability of collocation methods for the numerical solution of* $y'' = f(x, y)$, BIT 20, 215–222.
[13] Liniger, W. & Willoughby, R. A. (1970): *Efficient integration methods for stiff systems of ordinary differential equations*, SIAM J. Numer. Anal. 7, 47–66.
[14] Rosenbrock, H. H. (1963): *Some general implicit processes for the numerical solution of differential equations*, Comput. J. 5, 329–330.
[15] Watts, H. A. & Shampine, L. F. (1972): *A-stable block implicit one-step methods*, BIT 12, 252–266.
[16] Wright, K. (1970): *Some relationships between implicit Runge-Kutta, collocation and Lanczos* τ *methods, and their stability properties*, BIT 10, 217–227.

CHAPTER II

Parallel diagonally implicit Runge-Kutta-Nyström methods

published in: *Appl. Numer. Math.* **9** (1992), 111-131

Parallel diagonally implicit Runge–Kutta–Nyström methods *

P.J. van der Houwen and B.P. Sommeijer

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Nguyen huu Cong

Faculty of Mathematics, Mechanics and Informatics, University of Hanoi, Thuong dinh, Dong Da, Hanoi, Vietnam

Abstract

Van der Houwen, P.J., B.P. Sommeijer and Nguyen huu Cong, Parallel diagonally implicit Runge–Kutta–Nyström methods, Applied Numerical Mathematics 9 (1992) 111–131.

In this paper, we study diagonally implicit iteration methods for solving implicit Runge–Kutta–Nyström (RKN) methods on parallel computers. These iteration methods are such that in each step, the iterated method can be regarded as a diagonally implicit Runge–Kutta–Nyström method (DIRKN method). The number of stages of this DIRKN method depends on the number of iterations and may vary from step to step. Since a large number of these stages can be computed in parallel, and since the total number of stages can be kept small by a suitable choice of the parameters in the iteration process, the resulting variable-stage DIRKN methods are efficient on parallel computers. By using implicit Runge–Kutta–Nyström methods with high stage order, the phenomenon of order reduction exhibited in many problems with large Lipschitz constants does not deteriorate the accuracy of these variable-stage DIRKN methods. By a number of numerical experiments the superiority of the parallel iterated RKN methods over sequential DIRKN methods from the literature is demonstrated.

Keywords. Diagonally implicit Runge–Kutta–Nyström methods, predictor–corrector methods, parallelism.

1. Introduction

Consider the initial-value problem for systems of special second-order, ordinary differential equations (ODEs) of dimension d:

$$\begin{aligned} &y''(t) = f(y(t)), \qquad y(t_0) = y_0, \quad y'(t_0) = y_0', \\ &y : \mathbb{R} \to \mathbb{R}^d, \qquad f : \mathbb{R}^d \to \mathbb{R}^d, \qquad t_0 \leqslant t \leqslant t_{\text{end}}. \end{aligned} \tag{1.1}$$

* These investigations were supported by the University of Amsterdam who provided the third author with a research grant for spending a total of two years in Amsterdam.

0168-9274/92/$05.00 © 1992 – Elsevier Science Publishers B.V. All rights reserved

Important examples from this class of problems originate from structural mechanics. Such problems possess periodic solution components with frequencies ranging from small to large of which the lower harmonics are of interest and the higher harmonics are not, that is, only the solution components corresponding to eigenvalues of the Jacobian matrix $\partial f/\partial y$ close to the origin are of interest. In such cases, the ideal method would be a method without dissipation of the lower harmonics (i.e., nonempty periodicity interval), high order of dispersion, and damping of the higher harmonics. The presence of unwanted high harmonics (a form of stiffness) may considerably reduce the order at the step points (henceforth, this classical order will be called *step point order*). In many stiff problems, it is the stage order that determines the accuracy, rather than the step point order (cf. [2]). In order to diminish the effect of order reduction we need methods that have, in addition to a high step point order and the property of A-stability, a high stage order. We remark that A-stability for second-order problems is sometimes referred to as R-stability (cf. [17]).

In this paper, we consider integration methods based on iteration of fully implicit Runge–Kutta–Nyström (RKN) methods of collocation type. Such RKN methods possess the largest possible stage order, so that we automatically achieve high stage orders if the RKN method is solved sufficiently accurate. Furthermore, after only a few iterations, the step point order of the iterated method equals that of the underlying implicit RKN method. Since there are A-stable RKN methods available of arbitrarily high step point order, the iterated methods possess the requirements stated above. For an extensive set of suitable RKN methods with high stage orders we refer to [9].

In Section 2, we shall investigate *diagonal-implicit* iteration methods for solving the implicit relations in the RKN method. Such iteration methods possess the same degree of implicitness as diagonally implicit Runge–Kutta–Nyström methods (DIRKN methods). In fact, after a finite number of iterations, they belong to the class of DIRKN methods. We remark that the step point order p of these DIRKN methods can be made arbitrarily high by iterating an RKN method with step point order p, where p is sufficiently large. Hence, the restriction $p \leqslant 4$ which applies to the DIRKN methods available in the literature (see Section 1.2) is easily relaxed. Adopting the terminology used for iterating implicit linear multistep methods, we shall call the underlying implicit RKN method the *corrector* and the method used for starting the iteration the *predictor* (which are discussed in Sections 1.1 and 2.3, respectively). The iteration process will be called *predictor–corrector* (PC) method.

The number of stages of this PC method increases with the number of iterations and may vary from step to step depending on the convergence behaviour. Because of the nature of diagonal-implicit PC methods, a large number of the stages of the resulting variable-stage DIRKN method can be computed in parallel, so that the number of stages that have to be computed sequentially is substantially reduced when implemented on multi-processor computers. A second advantage is that only one LU-decomposition per processor is required. Hence, the method can be regarded as a *singly-implicit* DIRKN method (SDIRKN method). Thirdly, we shall reduce the number of iterations per step by a suitable choice of the parameters in the iteration process (to be discussed in Section 2.5). In this paper, our approach of choosing the iteration parameters is based on the minimization of the spectral radius of the stage vector iteration matrix. For a number of RKN correctors generated by collocation-based RK methods, we have calculated the iteration parameters with this minimizing property. However, fast convergence of the PC iteration is useless if the overall stability is insufficient. Therefore, from

the various PC methods, we selected (in Section 2.6) those methods that are A-stable for a minimal number of iterations per step. Finally, the use of collocation-based corrector methods guarantees high stage order, so that the phenomenon of order reduction, exhibited in many problems with large Lipschitz constants, does not deteriorate the accuracy of the methods.

By a number of numerical examples, it is demonstrated (see Section 3) that the high-order parallel SDIRKN methods proposed in this paper are by far superior to the sequential SDIRKN methods from the literature.

Finally, in Section 4, we briefly summarize the main results of this paper and discuss some possible extensions.

1.1. RKN methods

We consider RKN correctors of the form

$$\begin{aligned} y_{n+1} &= y_n + hy'_n + b_0 h^2 f(y_n) + h^2 \sum_{i=1}^{k} b_i f(Y_i), \\ y'_{n+1} &= y'_n + d_0 hf(y_n) + h \sum_{i=1}^{k} d_i f(Y_i), \\ Y_i &= y_n + c_i hy'_n + a_i h^2 f(y_n) + h^2 \sum_{j=1}^{k} a_{ij} f(Y_j), \quad i = 1, \ldots, k, \end{aligned} \tag{1.2}$$

or using the Butcher array notation (cf. e.g., [5]),

$$\begin{array}{c|cc} 0 & 0 & \mathbf{0}^{\mathrm{T}} \\ \boldsymbol{c} & \boldsymbol{a} & A \\ \hline & b_0 & \boldsymbol{b}^{\mathrm{T}} \\ & d_0 & \boldsymbol{d}^{\mathrm{T}} \end{array}, \tag{1.3}$$

where $\boldsymbol{a} = (a_i)$, $\boldsymbol{b} = (b_i)$, $\boldsymbol{c} = (c_i)$ and $\boldsymbol{d} = (d_i)$ are k-dimensional vectors, $A = (a_{ij})$ is a k by k matrix and $\mathbf{0}$ is a k-dimensional vector with zero entries. We always assume that the matrix A is nonsingular. Scheme (1.2) presents an $(s = k + 1)$-stage RKN method requiring k *implicit* stages and one explicit stage. In the case where $\boldsymbol{a}$, b_0 and d_0 vanish, the explicit stage is not needed and (1.2) reduces to the general $(s = k)$-stage RKN method with s implicit stages. For a discussion of the order of accuracy p and the stage order r of RKN methods, we refer to the literature (e.g., [4,9]).

It will be assumed that the eigenvalues of the Jacobian matrix $\partial f/\partial y$ in (1.1) are negative. This means that the integration step should satisfy the stability condition

$$h^2 \leqslant \frac{\beta_{\mathrm{stab}}}{\rho(\partial f/\partial y)}, \tag{1.4}$$

where $\rho(\partial f/\partial y)$ is the spectral radius of the Jacobian matrix $\partial f/\partial y$ and β_{stab} denotes the stability boundary of the RKN method. Thus, if we have a stiff problem where $\rho(\partial f/\partial y)$ is extremely large, then we should apply an A-stable RKN method, i.e., $\beta_{\mathrm{stab}} = \infty$. Unfortunately, the RKN methods with maximal stage order possess finite stability boundaries (cf. [10,9]). In

this connection, we remark that for certain classes of problems it is possible to use non-A-stable RKN methods for stiff problems by preconditioning the equation in (1.1). Then, instead of integrating (1.1), we integrate the equation (see [9])

$$\begin{aligned} &y''(t) = g(y(t)), \qquad g(y) := T^{-1}(J_n) f(y), \\ &y(t_n) = y_n, \qquad y'(t_n) = y_n', \qquad t_n \leqslant t \leqslant t_n + h, \\ &T(x) := 1 + \varepsilon(-x)^{\sigma}, \qquad \sigma := \lfloor (p+1)/2 \rfloor, \qquad J_n := \frac{\partial f(y_n)}{\partial y}, \end{aligned} \tag{1.5}$$

where p is the order of the RKN method and ε is a small parameter. The advantage is that, irrespective the size of the (negative) eigenvalue interval of $\partial f/\partial y$, the eigenvalues of $\partial g/\partial y$ are in a finite interval $[-\rho^*, 0]$, with

$$\rho^* := \frac{\sigma - 1}{\sigma[(\sigma - 1)\varepsilon]^{1/\sigma}}. \tag{1.6}$$

Hence, for the preconditioned equation (1.5) the stability condition (1.4) can be written as

$$\varepsilon \geqslant \frac{1}{\sigma - 1}\left(\frac{\sigma - 1}{\sigma \beta_{\text{stab}}}\right)^{\sigma} h^{2\sigma}, \tag{1.7}$$

where h denotes the step one wants to use. This condition shows that ε can be chosen of order $O(h^{2\sigma})$, so that (1.5) can be interpreted as a perturbed problem in which the perturbation is of order 2σ in h, that is, at least of order p.

In this paper, we shall concentrate on the iteration of A-stable RKN correctors. However, we shall present all formulas for equation (1.5), so that the use of non-A-stable RKN correctors is included in the subsequent analysis (notice that by setting $\varepsilon = 0$, we recover the original equation (1.1)). In a future paper, we intend to study the performance of non-A-stable RKN correctors with increased stage order.

1.2. Sequential SDIRKN methods from the literature

Although the total volume of arithmetic operations of the methods constructed in this paper is considerably larger than that of SDIRKN methods from the literature, matters are different when *parallel* computers are used. As we shall see in Section 2.1, many of the stages of the new methods can be performed in parallel, thus reducing the *effective* (or, *sequential*) run time to such an extent that it is comparable to that of SDIRKN methods on *sequential* machines. In order to facilitate a comparison of our parallel methods with already available sequential SDIRKN methods, we shall list a few of such SDIRKN methods from the literature.

Firstly, we remark that SDIRKN methods can be generated starting from SDIRK methods for first-order ODEs. Writing (1.1) in first-order form and application of an SDIRK method straightforwardly yields an SDIRKN method. Such methods will be called *indirect* SDIRKN methods. In particular, we mention the two-stage and three-stage A-stable SDIRKN methods of orders $p = 3$ and $p = 4$, respectively, based on the SDIRK methods of Nørsett [13]. These indirect methods will be denoted by Nørsett$_2$ and Nørsett$_3$, the subscript referring to the number of implicit stages per step. Since these methods do not possess an explicit stage, they

have vanishing $\boldsymbol{a}$, b_0 and d_0. Therefore, their Butcher arrays will be presented in the condensed form

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^{\mathrm{T}} \\ & \boldsymbol{d}^{\mathrm{T}} \end{array}. \tag{1.3'}$$

Using this format, the indirect Nørsett methods are now defined by

$$\begin{array}{c|cc} \lambda & \lambda^2 & 0 \\ 1-\lambda & 2\lambda(1-2\lambda) & \lambda^2 \\ \hline & \frac{1}{2}(1-\lambda) & \frac{1}{2}\lambda \\ & \frac{1}{2} & \frac{1}{2} \end{array}, \qquad \begin{array}{c|ccc} \xi & \xi^2 & 0 & 0 \\ \frac{1}{2} & \xi(1-2\xi) & \xi^2 & 0 \\ 1-\xi & 8\xi^2-3\xi+\frac{1}{2} & 2\xi(1-4\xi) & \xi^2 \\ \hline & 5\xi\eta-\xi-\eta+\frac{1}{2} & -6\xi\eta+\xi+\eta & \xi\eta \\ & \eta & 1-2\eta & \eta \end{array},$$

where

$$\lambda := \frac{3+\sqrt{3}}{6}, \qquad \xi := \frac{3+2\sqrt{3}\,\cos(\pi/18)}{6}, \qquad \eta := \frac{1}{6(1-2\xi)^2}.$$

Furthermore, we mention the indirect SDIRKN method generated by the third-order A-stable SDIRK method of Burrage [1]. This four-stage method has the special property that its order of B-convergence equals 3 for semi-linear problems. In the format (1.3'), its Butcher array reads

$$\begin{array}{r|rrrr} 0.7886751346 & 0.6220084679 & & & \\ 3.1742957030 & 3.7629592451 & 0.6220084679 & & \\ -0.0195951646 & -1.2749253740 & 0.0 & 0.6220084679 & \\ 1.0830184350 & 0.7564996127 & -0.0739506877 & -0.2182664410 & 0.6220084679 \\ \hline & -0.1353633836 & -0.0473517944 & 0.2862835400 & 0.3964316380 \\ & 0.0763188000 & -0.0301592919 & 0.4511853166 & 0.5026551753 \end{array}.$$

This method will be denoted by B_4.

In addition to the aforementioned indirect SDIRKN methods, we mention two *direct* SDIRKN methods. By "direct" we mean that they do not originate from an SDIRK method for first-order ODEs, but are constructed directly for the special second-order equation (1.1). In [17], Sharp, Fine and Burrage proposed two-stage and three-stage A-stable *direct* SDIRKN methods. In the form (1.3'), their Butcher arrays are given by

$$\begin{array}{c|cc} \frac{17}{14} & \frac{289}{392} & \\ \frac{23}{60} & -\frac{234179}{352800} & \frac{289}{392} \\ \hline & -\frac{21}{698} & \frac{185}{349} \\ & \frac{49}{349} & \frac{300}{349} \end{array}, \qquad \begin{array}{c|ccc} \frac{3}{5} & \frac{9}{50} & & \\ \frac{9}{10} & \frac{9}{40} & \frac{9}{50} & \\ \frac{6}{37} & \frac{234657}{1266325} & -\frac{891891}{2532650} & \frac{9}{50} \\ \hline & \frac{115}{729} & \frac{55}{2457} & \frac{42439}{132678} \\ & \frac{575}{1458} & \frac{550}{2457} & \frac{50653}{132678} \end{array}.$$

These methods have step point orders $p = 3$ and $p = 4$, respectively, and possess an increased order of dispersion which makes these methods highly accurate for oscillatory problems. They will be denoted by SFB_2 and SFB_3.

2. Diagonal-implicit PC methods

We shall construct integration methods by diagonal-implicit PC iteration of fully implicit RKN methods. Thus, assuming that in (1.2) the matrix $A=(a_{ij})$ is a full matrix, we have to find the solution of the equation for the stage vector $Y=(Y_i)$. Our aim is to construct solution methods that run fast on parallel computers. In the case where all eigenvalues of the Jacobian matrix are close to the origin, the stage vector equation in (1.2) can be solved by fixed point iteration which is well suited for implementation on parallel computers. For first-order ODEs this has been discussed in [14,11,7]. However, if the problem is "stiff" (by which we mean that $\partial g/\partial y$ also has negative eigenvalues of large modulus), then fixed point iteration would dictate very small stepsizes in order to get convergence. Therefore, we consider a more powerful class of parallel iteration processes which leads to the same degree of implicitness as occurring in SDIRKN methods. These processes are similar to the *stiff iteration method* applied in [8] for solving the stage vector equation associated with RK methods for first-order ODEs. In order to include RKN correctors that are not A-stable, the analysis will be presented for the preconditioned problem (1.5) (recall that (1.5) reduces to the original problem (1.1) if ε tends to zero).

2.1. Iteration of the stage-vector equation

Let $Y_i^{(\mu)}$ denote the μth iterate to Y_i, and define

$$\begin{aligned} &X_i := Y_i - x_i, \qquad X_i^{(\mu)} := Y_i^{(\mu)} - x_i, \\ &x_i := y_n + c_i h y_n' + a_i h^2 g(y_n), \end{aligned} \qquad i = 1,\dots,k. \tag{2.1}$$

Following [6] we shall compute iterates $X_i^{(\mu)}$, rather than the iterates $Y_i^{(\mu)}$, because the quantities $X_i^{(\mu)}$ are of smaller magnitude and are therefore less sensitive to rounding errors. In terms of X_i and x_i, the stage vector equation in (1.2) reads

$$X_i = h^2 \sum_{j=1}^{k} a_{ij} g(X_j + x_j), \quad i = 1,\dots,k. \tag{1.2'}$$

For each of these equations, we define the iteration process (cf. [8])

$$\begin{aligned} X_i^{(\mu)} - \delta_i h^2 g(X_i^{(\mu)} + x_i) = X_i^{(\mu-1)} - \delta_i h^2 g(X_i^{(\mu-1)} + x_i) \\ - \omega_\mu \left[X_i^{(\mu-1)} - h^2 \sum_{j=1}^{k} a_{ij} g(X_j^{(\mu-1)} + x_j) \right], \end{aligned} \tag{2.2}$$

where $i = 1,\dots,k$ and $\mu = 1,\dots,m$. Here, the ω_μ are relaxation parameters and the δ_i are iteration parameters which are assumed to be positive. Notice that the $X_i^{(\mu)}$ are *implicitly* defined in (2.2). This is a consequence of the introduction of the δ_i-parameters, and enables us to integrate stiff equations. In order to start the iteration (2.2), we need a predictor to compute the initial approximations $X_i^{(0)}$. The choice of a suitable predictor will be discussed in Section 2.3.

Evidently, if (2.2) converges, then $X_i^{(\mu)}$ converges to X_i. Since the k systems that are to be solved in each iteration of (2.2) can be solved *in parallel* and each has a dimension equal to that

of the system of ODEs, the iteration process (2.2) is on a k-processor computer of the same computational complexity as an m-stage SDIRKN method on a one-processor computer.

We remark that, if nonzero values for ε in (1.5) are used, then the implementation of the iteration formula (2.2) can be simplified by premultiplying with the matrix $T(J_n)$:

$$T(J_n)\boldsymbol{X}_i^{(\mu)} - \delta_i h^2 \boldsymbol{f}\big(\boldsymbol{X}_i^{(\mu)} + \boldsymbol{x}_i\big) = T(J_n)\boldsymbol{X}_i^{(\mu-1)} - \delta_i h^2 \boldsymbol{f}\big(\boldsymbol{X}_i^{(\mu-1)} + \boldsymbol{x}_i\big) - \omega_\mu \left[T(J_n)\boldsymbol{X}_i^{(\mu-1)} - h^2 \sum_{j=1}^{k} a_{ij} \boldsymbol{f}\big(\boldsymbol{X}_j^{(\mu-1)} + \boldsymbol{x}_j\big) \right]. \tag{2.2'}$$

This recursion shows that the preconditioning hardly complicates the form of the implicit relations to be solved.

2.1.1. Definition of the step values

Suppose that we adopt $\boldsymbol{Y}_i^{(m)} = \boldsymbol{X}_i^{(m)} + \boldsymbol{x}_i$ as a sufficiently accurate approximation to the exact stage vector solutions $\boldsymbol{Y}_i$ of the corrector (1.2). Then, the most natural way to approximate the step values $\boldsymbol{y}_{n+1}$ and $\boldsymbol{y}'_{n+1}$ in (1.2) defines the values according to the formulas (cf. [8])

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{y}'_n + b_0 h^2 \boldsymbol{g}(\boldsymbol{y}_n) + h^2 \sum_{i=1}^{k} b_i \boldsymbol{g}\big(\boldsymbol{Y}_i^{(m)}\big), \\ \boldsymbol{y}'_{n+1} &= \boldsymbol{y}'_n + d_0 h \boldsymbol{g}(\boldsymbol{y}_n) + h \sum_{i=1}^{k} d_i \boldsymbol{g}\big(\boldsymbol{Y}_i^{(m)}\big) \end{aligned} \tag{2.3}$$

(in order to avoid confusion, we shall from now on denote the corrector solution values obtained from $\boldsymbol{y}_n$ and $\boldsymbol{y}'_n$ by $\boldsymbol{u}_{n+1}$ and $\boldsymbol{u}'_{n+1}$). However, the presence of the righthand side evaluations in these formulas may give rise to loss of accuracy in the case of stiff problems (cf. [16]). This difficulty can be overcome by applying a similar approach as proposed in [6] for the implementation of implicit RK methods. For simplicity, we describe this approach for the scalar equation $y'' = g(y)$. Defining $\boldsymbol{Y} = (Y_i)$ and $\boldsymbol{G} = (g(Y_i))$, the corrector (1.2) can be written in the form

$$u_{n+1} = y_n + hy'_n + b_0 h^2 g(y_n) + h^2 \boldsymbol{b}^{\mathrm{T}} \boldsymbol{G}, \qquad u'_{n+1} = y'_n + d_0 h g(y_n) + h \boldsymbol{d}^{\mathrm{T}} \boldsymbol{G},$$
$$\boldsymbol{G} = h^{-2} A^{-1} \big[\boldsymbol{Y} - \boldsymbol{e} y_n - \boldsymbol{c} h y'_n - \boldsymbol{a} h^2 g(y_n) \big],$$

provided $\boldsymbol{A}$ is nonsingular. This representation shows that we can eliminate the righthand side evaluations and that u_{n+1} and u'_{n+1} can be expressed solely in terms of the stage vector $\boldsymbol{Y}$. Now we will compute y_{n+1} and y'_{n+1} according to these formulas with $\boldsymbol{Y}$ replaced by $\boldsymbol{Y}^{(m)}$. Returning to systems of ODEs and to the notation $\boldsymbol{X}_i^{(m)}$, we obtain

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{y}'_n + b_0 h^2 \boldsymbol{g}(\boldsymbol{y}_n) + \sum_{i=1}^{k} \alpha_i \boldsymbol{X}_i^{(m)}, \\ \boldsymbol{y}'_{n+1} &= \boldsymbol{y}'_n + d_0 h \boldsymbol{g}(\boldsymbol{y}_n) + h^{-1} \sum_{i=1}^{k} \beta_i \boldsymbol{X}_i^{(m)}, \end{aligned} \tag{2.4a}$$

where α_i and β_i are the components of the vectors

$$\boldsymbol{\alpha} := \boldsymbol{b}^{\mathrm{T}}A^{-1}, \qquad \boldsymbol{\beta} := \boldsymbol{d}^{\mathrm{T}}A^{-1}. \tag{2.4b}$$

In many cases the corrector is of so-called "stiffly accurate" type, which means that it satisfies $c_k = 1$, $b_0 = a_k$ and $\boldsymbol{b}^{\mathrm{T}}A^{-1} = \boldsymbol{e}_k^{\mathrm{T}}$ (see e.g. [2,6]). In such cases, the step value $\boldsymbol{u}_{n+1}$ produced by the corrector is given by the last component of the stage vector, i.e., by $\boldsymbol{Y}_k$. Accordingly, in case of a stiffly accurate corrector, the final approximation $\boldsymbol{y}_{n+1}$ at the steppoints is obtained by taking the last component of the iterated analogue, i.e., $\boldsymbol{Y}_k^{(m)}$. In terms of the iterate $\boldsymbol{X}_k^{(m)}$, $\boldsymbol{y}_{n+1}$ is defined by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{y}'_n + b_0 h^2 \boldsymbol{g}(\boldsymbol{y}_n) + \boldsymbol{X}_k^{(m)}. \tag{2.4'}$$

2.2. The iteration error

We shall say that the *order of the iteration error* of the PC method {(2.1), (2.2), (2.4)} equals q if

$$\boldsymbol{u}_{n+1} - \boldsymbol{y}_{n+1} = \mathrm{O}(h^{q+1}), \qquad \boldsymbol{u}'_{n+1} - \boldsymbol{y}'_{n+1} = \mathrm{O}(h^{q+1}), \tag{2.5}$$

where $(\boldsymbol{u}_{n+1}, \boldsymbol{u}'_{n+1})$ and $(\boldsymbol{y}_{n+1}, \boldsymbol{y}'_{n+1})$ denote the step values obtained from the values $(\boldsymbol{y}_n, \boldsymbol{y}'_n)$ by respectively solving the corrector equation exactly and by performing a finite number of iterations. The iteration error associated with {(2.1), (2.2), (2.4)} can be studied by applying it to the scalar test equation $y'' = \lambda y$, where λ runs through the eigenvalues of $\partial \boldsymbol{g}/\partial \boldsymbol{y}$. Defining the error

$$\varepsilon^{(\mu)} := \boldsymbol{X} - \boldsymbol{X}^{(\mu)}, \qquad \boldsymbol{X} := (X_i), \quad \boldsymbol{X}^{(\mu)} := (X_i^{(\mu)}), \tag{2.6}$$

we deduce from (2.2) that the iteration error (2.6) satisfies the recursion

$$\varepsilon^{(\mu)} = \left[I - \omega_\mu H(z)\right]\varepsilon^{(\mu-1)},$$

$$H(z) := \left[I - zD\right]^{-1}\left[I - zA\right], \quad z := \lambda h^2, \ \mu = 1, \ldots, m,$$

where D is the diagonal matrix with diagonal entries δ_i. Hence,

$$\varepsilon^{(m)} = P_m(H(z))\varepsilon^{(0)}, \qquad P_m(x) = \prod_{\mu=1}^{m} (1 - \omega_\mu x). \tag{2.7}$$

The matrix $P_m(H(z))$ will be called the *stage vector iteration matrix*.

In the following, we use the notation

$$\boldsymbol{w}_{n+1} := \begin{pmatrix} u_{n+1} \\ hu'_{n+1} \end{pmatrix}, \qquad \boldsymbol{v}_{n+1} := \begin{pmatrix} y_{n+1} \\ hy'_{n+1} \end{pmatrix}. \tag{2.8a}$$

In terms of these vectors, we can derive an error equation of the form

$$\boldsymbol{w}_{n+1} - \boldsymbol{v}_{n+1} = E_m(z)\boldsymbol{v}_n, \tag{2.8b}$$

where the matrix $E_m(z)$ is a 2 by 2 matrix determined by the RKN parameters and the matrix D. This matrix will be called the *iteration matrix of the diagonal-implicit PC method*. From the formulas (2.4) and (2.4′) for the step values it follows that

$$u_{n+1} - y_{n+1} = \boldsymbol{p}^{\mathrm{T}}P_m(H(z))\varepsilon^{(0)}, \qquad hu'_{n+1} - hy'_{n+1} = \boldsymbol{d}^{\mathrm{T}}A^{-1}P_m(H(z))\varepsilon^{(0)}, \tag{2.9}$$

where $\boldsymbol{p}^{\mathrm{T}} = \boldsymbol{b}^{\mathrm{T}}A^{-1}$ for nonstiffly accurate correctors, and $\boldsymbol{p}^{\mathrm{T}} = \boldsymbol{e}_k^{\mathrm{T}}$ for stiffly accurate correctors.

We shall first give an order result for the PC method. The actual choice of the predictor will be discussed in Section 2.3. The preceding considerations lead to the following theorem:

Theorem 2.1. *Let the predictor be of order* p^*, *i.e.*,

$$\varepsilon^{(0)} = \boldsymbol{X} - \boldsymbol{X}^{(0)} = \mathrm{O}(h^{p^*+1}).$$

Let

$$P_m(x) = (1-x)^{q^*} Q_{m-q^*}(x), \quad Q_{m-q^*}(1) \neq 0.$$

Then, for any choice of the matrix D, *the order* q *of the iteration error of the PC method* {(2.1), (2.2), (2.4)} *is given by* $q = 2q^* + p^* - 1$.

Proof. Since P_m has a zero at $x = 1$ of multiplicity q^*, it follows from (2.9) that for $z \to 0$

$$u_{n+1} - y_{n+1} = z^{q^*} Q_{m-q^*}(1) \boldsymbol{p}^{\mathrm{T}} (A - D)^{q^*} \mathrm{O}(h^{p^*+1}),$$

$$u'_{n+1} - y'_{n-1} = z^{q^*} Q_{m-q^*}(1) \boldsymbol{d}^{\mathrm{T}} A^{-1} (A - D)^{q^*} h^{-1} \mathrm{O}(h^{p^*+1}).$$

Recalling definition (2.5) and observing that $z = \mathrm{O}(h^2)$, the theorem easily follows. □

2.3. The predictor

In view of stability, an important property of the predictors is the degree of amplification of stiff components. Therefore, apart from the usual approach to choose an explicit predictor, we will also consider some implicit predictors. Notice that, as a consequence of this choice, the number of implicit relations to be solved per step is increased by one.

In Table 1 we have collected various possibilities for choosing the predictor. We remark that, in this paper, we confine our considerations to *one-step* predictors. Notice that these low-order predictors might be improved upon by using multistep predictors of higher order, since it is likely that these will result in fewer iterations. Observe that the predictors III and IV are of order 2, whereas the first two predictors are only of first order. Furthermore, we remark that the predictors II and IV have a strongly damping effect on the stiff components.

To compare the computational costs required by the various predictors, we also list the number of systems of dimension d to be solved in each step on each processor, and the number of sequential LU-decompositions (LUDs) per step. Since predictor IV needs an LUD of the matrix $I - \gamma_i h^2 \partial \boldsymbol{g}/\partial \boldsymbol{y}$ (to solve for $\boldsymbol{X}_i^{(0)}$) in addition to an LUD of $I - \delta_i h^2 \partial \boldsymbol{g}/\partial \boldsymbol{y}$ (needed in

Table 1
Survey of one-step predictors ($\gamma_i = \frac{1}{2}c_i^2$)

Predictor	$X_i^{(0)}$, $i = 1, \ldots, k$	p^*	Systems	LUDs
I	$-a_i h^2 \boldsymbol{g}(\boldsymbol{y}_n)$	1	m	1
II	$-a_i h^2 \boldsymbol{g}(\boldsymbol{y}_n) + \delta_i h^2 \boldsymbol{g}(\boldsymbol{X}_i^{(0)} + \boldsymbol{x}_i)$	1	$m+1$	1
III	$-a_i h^2 \boldsymbol{g}(\boldsymbol{y}_n) + h^2[\delta_i \boldsymbol{g}(\boldsymbol{X}_i^{(0)} + \boldsymbol{x}_i) + (\gamma_i - \delta_i)\boldsymbol{g}(\boldsymbol{y}_n)]$	2	$m+1$	1
IV	$-a_i h^2 \boldsymbol{g}(\boldsymbol{y}_n) + \gamma_i h^2 \boldsymbol{g}(\boldsymbol{X}_i^{(0)} + \boldsymbol{x}_i)$	2	$m+1$	2

each iteration of (2.2)), this predictor seems to be less attractive from a computational point of view.

The predictors listed in Table 1 are such that we can write

$$\varepsilon^{(0)} = \boldsymbol{X} - \boldsymbol{X}^{(0)} = \boldsymbol{k}_1(z)\, y_n + \boldsymbol{k}_2(z) h y_n', \tag{2.10}$$

where the vectors $\boldsymbol{k}_1(z)$ and $\boldsymbol{k}_2(z)$ are determined by the RKN parameters and, in case of the predictors II and III, also by the matrix D. The iteration matrix $E_m(z)$ in (2.8b) assumes the form

$$E_m(z) := \begin{pmatrix} \boldsymbol{p}^{\mathrm{T}} P_m(H(z))\boldsymbol{k}_1(z) & \boldsymbol{p}^{\mathrm{T}} P_m(H(z))\boldsymbol{k}_2(z) \\ \boldsymbol{d}^{\mathrm{T}} A^{-1} P_m(H(z))\boldsymbol{k}_1(z) & \boldsymbol{d}^{\mathrm{T}} A^{-1} P_m(H(z))\boldsymbol{k}_2(z) \end{pmatrix}. \tag{2.8c}$$

This matrix will be used in deriving the stability function of the PC methods (see Section 2.5.3).

2.4. The rate of convergence

Ideally, the overall rate of convergence should be based on some norm of the iteration matrix $E_m(z)$ for all z-values that are relevant for the problem (1.5). However, this would lead to iteration parameters that depend on the predictor and on m. This is an undesirable situation, since the number of iterations m is not known a priori and may vary from step to step. By observing that the entries of $E_m(z)$ are small if the magnitude of the stage vector iteration matrix $P_m(H(z))$ is small, we are led to minimize, in some sense, the magnitude of $P_m(H(z))$ for negative values of z. In this paper, we consider the case where the magnitude of $P_m(H(z))$ is estimated by its spectral radius. By minimizing the spectral radius of $P_m(H(z))$, the iteration parameters can be determined independently of the predictor and of the number of iterations m. Denoting the spectral radius of a matrix M by $\rho(M)$, we characterize the rate of convergence of the stage vector iteration by

$$\begin{aligned} &r_m(z) := = \sqrt[m]{\rho(P_m(H(z)))}\,, \qquad r_m := \operatorname*{Max}_{-\beta \leqslant z \leqslant 0} r_m(z), \\ &\boldsymbol{r} := (r_1, r_2, \ldots)^{\mathrm{T}}, \quad \beta := h^2 \rho^*, \end{aligned} \tag{2.11}$$

where ρ^* is the parameter occurring in (1.6).

Furthermore, we denote the spectrum of $H(z)$ by $\Lambda(H(z))$, and we define

$$\begin{aligned} &\rho(z) := \operatorname{Max}\{|\lambda - 1|: \lambda \in \Lambda(H(z))\}, \qquad \rho := \operatorname{Max}\{|\lambda - 1|: \lambda \in \Lambda(H)\}, \\ &\Lambda(H) := \{\Lambda(H(z)): -\beta \leqslant z \leqslant 0\}. \end{aligned} \tag{2.12}$$

2.5. Choice of iteration parameters

In the following subsections, we shall discuss a few special cases for choosing the relaxation parameters ω_μ and the matrix D. We start with a discussion of the *stiff iteration* approach which was investigated in [8] for solving implicit RK methods for first-order ODEs.

2.5.1. Stiff iteration

In this case the matrix D is such that $\Lambda(H(-\infty))$ is contained in a circle with minimal radius $\rho(-\infty)$ and centered at 1, and the relaxation parameters are all equal to 1, so that $r_m = \rho$. Stiff iteration preassumes that the corrector is A-stable, hence we set $\beta = \infty$ in (2.15). The following theorem holds for $k = 2$:

Theorem 2.2. *Let $k = 2$, then the following assertions hold for the stiff iteration method:*

(a) *if $\det(A) > 0$ and if either $\{a_{12}a_{21} \leqslant 0$ and $a_{22} \geqslant 0\}$ or $\{a_{11} > 0$ and $a_{22} \leqslant 0\}$, then there exists a matrix D with positive entries such that $\rho(-\infty) = 0$.*

(b) *if* (a) *holds, then one eigenvalue of $H(z)$ equals 1 for all z.*

(c) *if* (a) *holds and if $\mathrm{Tr}(A) > -2\,[\det(A)]^{1/2}$, then the eigenvalues of $H(z)$ are real and positive for all negative z.*

Proof. (a) For $k = 2$ the value of $\rho(-\infty)$ vanishes if the matrix $H(-\infty) - I = D^{-1}A - I$ has zero eigenvalues. This can be achieved by choosing

$$\delta_1 = \frac{\det(A)}{a_{22}}\left(1 \pm \sqrt{1 - a_{11}a_{22}/\det(A)}\,\right), \qquad \delta_2 = \frac{2\det(A) - \delta_1 a_{22}}{a_{11}}.$$

By an elementary calculation assertion (a) can now be verified.

(b) Assertion (b) is satisfied if there exists a vector $\boldsymbol{v}$, such that $H(z)\boldsymbol{v} = \boldsymbol{v}$ for all z, i.e., if $(I - zA)\boldsymbol{v} = (I - zD)\boldsymbol{v}$. This relation is true for all z if $D^{-1}A\boldsymbol{v} = \boldsymbol{v}$. Evidently, if (a) holds, then $D^{-1}A$ has only eigenvalues 1 which proves (b).

(c) Since the entries of $H(z)$ are real for all negative z, we deduce from (b) that $H(z)$ has real eigenvalues for $z < 0$. Hence, by showing that

$$\det(H(z)) = \det(I - zD)^{-1}\det(I - zA) = \det(I - zD)^{-1}\left[\det(A)z^2 - \mathrm{Tr}(A)z + 1\right]$$

is positive for $z < 0$, we can prove assertion (c). □

Table 2
Stiff matrices D and corresponding vectors r

Generating RK method	s	k	δ_1	δ_2	δ_3	δ_4	$\rho(-\infty)$	r
Radau IIA	2	2	1/18	1/2			0	$e/2$
Lobatto IIIA (≡ Newton–Cotes)	3	2	1/24	1/6			0	$e/3$
Lagrange with $c = (\frac{3}{4}, 1)^T$	3	2	3/32	1/6			0	$e/3$
Radau IIA	3	3	8417/16328	255/19799	1483/35645		0.0028	$0.77e$
Lobatto IIIA	4	3	754/7243	113/12480	999/13576		0.0007	$0.52e$
Newton–Cotes	4	3	125/8979	988/18531	85/729		0.0035	$0.52e$
Lagrange: $c = (\frac{7}{12}, \frac{5}{6}, 1)^T$	4	3	362/8683	605/7281	783/6628		0.0019	$0.53e$
Radau IIA	4	4	2625/7342	1225/7601	76/20731	71/10024	0.023	$0.81e$
Lobatto IIIA	5	4	3384/40409	25/5154	221/10255	134/3319	0.074	$0.58e$
Newton–Cotes	5	4	81/12772	493/20960	337/6661	921/10594	0.026	$0.64e$
Lagrange: $c = (\frac{1}{6}, \frac{7}{12}, \frac{11}{12}, 1)^T$	5	4	71/4500	105/9613	400/7807	1177/18717	0.016	$0.55e$

Table 3
Zarantonello matrices D and corresponding vectors $\boldsymbol{r}$

Generating RK method	s	k	δ_1	δ_2	δ_3	δ_4	$\rho(-\infty)$	$\boldsymbol{r}$
Radau IIA	2	2	1/24	3/8				$0.33\boldsymbol{e}$
Lobatto IIIA (≡ Newton–Cotes)	3	2	611/17603	347/2500				$0.20\boldsymbol{e}$
Lagrange $\boldsymbol{c}=(\frac{3}{4}, 1)^{\mathrm{T}}$	3	2	391/5000	139/1000				$0.20\boldsymbol{e}$
Radau IIA	3	3	453/2500	47/2500	547/2500		0.47	$0.64\boldsymbol{e}$
Lobatto IIIA	4	3	133/1250	1/100	431/5000		0.47	$0.47\boldsymbol{e}$
Newton–Cotes	4	3	57/5000	441/10000	971/10000		0.36	$0.42\boldsymbol{e}$
Lagrange with $\boldsymbol{c}=(\frac{7}{12}, \frac{5}{6}, 1)^{\mathrm{T}}$	4	3	43/1250	69/1000	578/5871		0.34	$0.44\boldsymbol{e}$
Radau IIA	4	4	2625/7342	1225/7601	76/20731	71/10024	0.023	$0.81\boldsymbol{e}$
Lobatto IIIA	5	4	1/10	1/200	1/50	7/200	0.561	$0.57\boldsymbol{e}$
Newton–Cotes	5	4	81/12772	493/20960	337/6661	921/10594	0.026	$0.64\boldsymbol{e}$
Lagrange: $\boldsymbol{c}=(\frac{1}{6}, \frac{7}{12}, \frac{11}{12}, 1)^{\mathrm{T}}$	5	4	71/4500	105/9613	400/7807	1177/18717	0.016	$0.55\boldsymbol{e}$

For $k > 2$, we did not succeed in deriving the optimal matrix D by analytical methods, so that we resorted to numerical search techniques. For a few RKN correctors generated by classical RK methods, Table 2 presents the entries of the matrices D that are optimal for stiff iteration (stiff matrices D). The given entries in this table (and in the subsequent tables) are rational approximations to the decimal numbers we found. Furthermore, we include the RKN correctors generated by the Lagrange methods with collocation vectors $\boldsymbol{c}=(\frac{3}{4}, 1)^{\mathrm{T}}$, $\boldsymbol{c}=(\frac{7}{12}, \frac{5}{6}, 1)^{\mathrm{T}}$ and $\boldsymbol{c}=(\frac{1}{6}, \frac{7}{12}, \frac{11}{12}, 1)^{\mathrm{T}}$ proposed in [8]. For all methods, we also list the vectors $\boldsymbol{r}$ as defined in (2.11).

2.5.2. Zarantonello iteration

Assuming that all relaxation parameters equal 1, the optimal choice of the set $\Lambda(H)$ is a circle centered at 1 with minimal radius ρ. This follows from a lemma of Zarantonello (cf. [18]), stating that the spectral radius of $P_m(H(z))$ is minimized if P_m has all its zeros at the center of the circle (with smallest radius) containing the eigenvalues of $H(z)$. We shall call this iteration mode *Zarantonello iteration*. As for stiff iteration, we have $r_m = \rho$, however, r_m is expected to be smaller.

A numerical search yields the results listed in Table 3. For the 4-stage Radau IIA and the 5-stage Newton–Cotes and Lagrange correctors we could not find a better D matrix than in the stiff case, so that the Zarantonello matrix D is identical with the stiff matrix D yielding identical convergence factors. In all other cases, Zarantonello iteration possesses considerably better convergence factors.

2.5.3. Chebyshev iteration

The PC method can be made more rapidly converging by a more sophisticated choice of the relaxation parameters ω_μ and the iteration parameters δ_i. The optimal choice of the relaxation parameters leads to a minimax problem for the polynomial $P_m(x)$ on the set $\Lambda(H)$. Such minimax problems have been extensively studied in the literature and can be solved by identifying the polynomial P_m in (2.7) with a shifted Chebyshev polynomial, the shift parameters being determined by the ellipsoidal region containing the complex set $\Lambda(H)$ (see [12]). We

shall consider this approach for the simplified case where the matrix D is such that $\Lambda(H)$ is contained in a real positive interval $[a, b]$. The optimal choice of P_m is then given by the polynomial (see e.g. [18])

$$P_m(x) = \frac{1}{\tau_m} T_m\left(\frac{b+a-2x}{b-a}\right), \qquad \tau_m := T_m\left(\frac{b+a}{b-a}\right), \tag{2.13}$$

where T_m denotes the first-kind Chebyshev polynomial of degree m. From (2.7) it follows that the corresponding relaxation parameters are the inverses of the zeros of the polynomial (2.13), i.e.,

$$\omega_\mu = \frac{2}{b+a-(b-a)\cos\left(\dfrac{(2\mu-1)\pi}{2m}\right)}, \quad \mu = 1, \ldots, m.$$

Since P_m is bounded by $1/\tau_m$, we may write

$$r_m := \sqrt[m]{\frac{1}{\tau_m}} \approx \sqrt[m]{2}\,\frac{b-a}{2(b+a)} \in \frac{b-a}{b+a}\left[\tfrac{1}{2}, 1\right] \quad \text{as } a \to b. \tag{2.14}$$

Evidently, the Chebyshev approach will be more rapidly converging as b/a is closer to 1, hence we determined D such that b/a is minimal (notice that $b \geqslant 1$). The corresponding iteration method will be called *Chebyshev iteration*.

From Theorem 2.2 it follows that under the conditions of part (a) of the theorem, the matrices D corresponding to stiff iteration can also be used for Chebyshev iteration. It turns out that the conditions of part (a) are fulfilled by a number of RKN correctors generated by classical RK collocation methods for first-order ODEs (for these correctors, we have $\beta_{\text{stab}} = \infty$). Moreover, we found that for these correctors the corresponding matrices D minimize the value of b/a. Hence:

Corollary 2.3. *For $k = 2$ the matrices D corresponding to stiff iteration are optimal for Chebyshev iteration.*

Table 4 presents the matrices D that are optimal for Chebyshev iteration (Chebyshev matrices D) and the numbers $r_1, r_2, \ldots, r_\infty$ as defined in (2.14). A comparison with Tables 2 and 3 reveals that the convergence of Chebyshev stage vector iteration should be substantially faster than that of stiff and Zarantonello iteration. A number of experiments where the rates of convergence in a single step were considered, confirmed this conclusion. However, when the global result of a whole integration process is considered, it turned out that Chebyshev iteration is by far inferior to stiff and Zarantonello iteration. This is illustrated in the following example.

Example 2.4. Consider the model problem (see Kramarz [10]):

$$y''(t) = \begin{pmatrix} 2498 & 4998 \\ -2499 & -4999 \end{pmatrix} y(t), \qquad y(0) = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \quad y'(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad 0 \leqslant t \leqslant 100, \tag{2.15}$$

Table 4
Chebyshev matrices D and corresponding vectors r

Generating IRK method	s	k	δ_1	δ_2	δ_3	δ_4	$[a,b]$	$r^T := (r_1, r_2, \ldots, r_\infty)^T$
Radau IIA	2	2	1/18	1/2			[1/2,1]	(0.34, 0.25,...,0.18)
Lobatto IIIA ($\equiv$ Newton–Cotes)	3	2	1/24	1/6			[2/3,1]	(0.20, 0.15,...,0.11)
Lagrange with $c = (\frac{3}{4}, 1)^T$	3	2	3/32	1/6			[2/3,1]	(0.20, 0.15,...,0.11)
Radau IIA	3	3	18/125	7/500	11/50		[0.33,1.48]	(0.63, 0.50,...,0.36)
Lobatto IIIA	4	3	7/40	7/500	31/200		[0.27,1.00]	(0.57, 0.44,...,0.32)
Newton–Cotes	4	3	1/50	81/1000	93/500		[0.29,1.00]	(0.55, 0.42,...,0.30)
Lagrange: $c = (\frac{7}{12}, \frac{5}{6}, 1)^T$	4	3	1/25	17/200	1/8		[0.41, 1.74]	(0.62, 0.49,...,0.35)
Radau IIA	4	4	141/1000	7/1000	7/125	31/200	[0.21,1.49]	(0.75, 0.63,...,0.46)
Lobatto IIIA	5	4	1/20	9/1000	113/1000	9/50	[0.15,1.00]	(0.73, 0.61,...,0.44)
Newton–Cotes	5	4	1/125	43/1250	167/2000	4/25	[0.18,1.38]	(0.77, 0.65,...,0.47)
Lagrange: $c = (\frac{1}{6}, \frac{7}{12}, \frac{11}{12}, 1)^T$	5	4	29/1000	1/50	53/500	33/250	[0.20,1.00]	(0.66, 0.53,...,0.38)

with exact solution $y(t) = (2\cos(t), -\cos(t))^T$. For the indirect two-stage Radau IIA corrector Table 5 lists the number of minimal correct digits

$$\mathrm{NCD}(h) := -\log(\| \textit{global error at the endpoint of the integration interval} \|_\infty)$$

obtained for a few values of h and m. Negative NCD-values are indicated by $*$. This table shows the inferiority of Chebyshev iteration. Since the matrices D in the stiff and Chebyshev iteration mode of the indirect two-stage Radau IIA corrector are identical (see Corollary 2.3), the poor performance of Chebyshev iteration is apparently caused by the choice of the relaxation parameters.

The explanation of the poor overall performance of Chebyshev iteration is that, in spite of the rapid Chebyshev convergence in each step, the stability of the integration process requires

Table 5
NCD-values for problem (2.15) obtained by the PC method with predictor I and indirect two-stage Radau IIA corrector

Iteration mode	h	$m=2$	$m=3$	$m=4$	$m=5$	...	$m=\infty$
Stiff	$\frac{1}{10}$	2.5	2.6				2.6
	$\frac{1}{20}$	3.4	3.5				3.5
	$\frac{1}{40}$	4.4					4.4
Zarantonello	$\frac{1}{10}$	2.5	2.6				2.6
	$\frac{1}{20}$	3.5					3.5
	$\frac{1}{40}$	4.4					4.4
Chebyshev	$\frac{1}{10}$	*	*	0.1	0.9	...	2.6
	$\frac{1}{20}$	*	*	*	0.3	...	3.5
	$\frac{1}{40}$	*	*	*	*	...	4.4

many more iterations per step than required by the convergence criterion. To see the reasons for this phenomenon we have to define the *stability function* for diagonal-implicit PC methods. The RKN corrector satisfies the relation (cf. [9])

$$\boldsymbol{w}_{n+1} = M(z)\boldsymbol{v}_n,$$

$$M(z) := \begin{pmatrix} 1 + zb_0 + z\boldsymbol{b}^{\mathrm{T}}(I - Az)^{-1}[\boldsymbol{e} + z\boldsymbol{a}] & 1 + z\boldsymbol{b}^{\mathrm{T}}(I - Az)^{-1}\boldsymbol{c} \\ zd_0 + z\boldsymbol{d}^{\mathrm{T}}(I - Az)^{-1}[\boldsymbol{e} + z\boldsymbol{a}] & 1 + z\boldsymbol{d}^{\mathrm{T}}(I - Az)^{-1}\boldsymbol{c} \end{pmatrix}. \tag{2.16}$$

On substitution into (2.8b) we obtain

$$\boldsymbol{v}_{n+1} = [M(z) - E_m(z)]\boldsymbol{v}_n. \tag{2.17}$$

We shall call the matrix $M(z) - E_m(z)$ the *stability matrix* of the iterated RKN method and its spectral radius the *stability function* $R_m(z)$, i.e.,

$$R_m(z) := \rho([M(z) - E_m(z)]). \tag{2.18}$$

From (2.16) it follows that $M(z)$ approaches a matrix with a double unit eigenvalue for $z \to 0$. As a consequence, the eigenvalues of the stability matrix $M(z) - E_m(z)$ may easily move outside the unit circle, unless the entries of $E_m(z)$ are close to zero as $z \to 0$. The definition of $E_m(z)$ strongly suggests choosing all zeros of the polynomial $P_m(x)$ at $x = 1$, i.e., all relaxation parameters equal to 1. In order to illustrate that unit relaxation parameters improve the performance dramatically, we repeated the experiment in Example 2.4 by iterating the indirect three-stage Radau IIA corrector using relaxation parameters equal to 1 together with the Chebyshev matrix D (*stationary Chebyshev iteration*).

Example 2.5. Table 6 compares the Chebyshev and stationary Chebyshev mode of the indirect three-stage Radau IIA corrector for problem (2.15). The superiority of the stationary Chebyshev mode over the "true" Chebyshev mode is evident.

2.6. Selection of methods

Since stability plays such a crucial role in the overall performance of the PC methods, we have computed (numerically) the minimal value of m such that the iteration method is stable

Table 6
NCD-values for problem (2.15) obtained by the PC method with predictor I and indirect three-stage Radau IIA corrector

Iteration mode	h	$m = 2$	$m = 3$	$m = 4$	$m = 5$	...	$m = \infty$
Chebyshev	$\frac{1}{10}$	*	*	*	*	...	6.6
	$\frac{1}{20}$	*	*	*	*	...	8.1
	$\frac{1}{40}$	*	*	*	*	...	9.6
Stationary	$\frac{1}{10}$	*	*	*	6.6		6.6
Chebyshev	$\frac{1}{20}$	4.6	7.7	8.1			8.1
	$\frac{1}{40}$	5.9	9.4	9.6			9.6

Table 7
Values of m_1 for Zarantonello and stationary Chebyshev iteration

Generating RK method	k	$r=s$	Zarantonello iteration		Stationary Chebyshev iteration	
			Explicit Predictor	Implicit Predictor	Explicit Predictor	Implicit Predictor
Radau IIA	2	2	4	**2**	**2**	**2**
Lobatto IIIA (≡ Newton–Cotes)	2	3	> 10	> 10	7	7
Lagrange with $c=(\frac{3}{4}, 1)^T$	2	3	3	4	**2**	3
Radau IIA	3	3	8	5	7	**4**
Lobatto IIIA	3	4	> 10	> 10	> 10	> 10
Newton–Cotes	3	4	> 10	> 10	> 10	> 10
Lagrange: $c=(\frac{7}{12}, \frac{5}{6}, 1)^T$	3	4	6	6	6	7
Radau IIA	4	4	> 10	> 10	> 10	> 10
Lobatto IIIA	4	5	> 10	> 10	> 10	> 10
Newton–Cotes	4	5	> 10	> 10	> 10	> 10
Lagrange: $c=(\frac{1}{6}, \frac{7}{12}, \frac{11}{12}, 1)^T$	4	5	**7**	8	> 10	> 10

for all z in the interval $[-\beta,0]$ and for all m equal to or greater than this value. Let us denote this critical value of m by m_0 and let m_1 denote the minimal number of systems (of dimension d) that are to be solved per step and per processor such that the PC method is stable. From Table 1 it follows that $m_1 = m_0$ for the explicit predictor I, and $m_1 = m_0 + 1$ for the implicit predictors II, III, and IV. In Table 7, the values of m_1 are listed for a number of RK-generated RKN correctors using the explicit predictor I and the implicit predictor II. For each k, the minimal values are indicated in bold face.

This table shows that for $k=2$ there are four combinations of predictor, corrector and iteration mode with a minimal m_1-value. From these combinations we have chosen the Lagrange-based method because the stage order r of the indirect Lagrange corrector is higher than that of the indirect Radau corrector. For $k=3$ and $k=4$ there is just one "optimal" combination. Thus, we are led to the following three optimal A-stable combinations:

$$\begin{aligned}&\text{Explicit–Lagrange–Chebyshev}\\&\quad\text{with at least 2 implicit sequential stages } (\mathrm{ELC}_2)\\&\text{Implicit–Radau IIA–Chebyshev}\\&\quad\text{with at least 4 implicit sequential stages } (\mathrm{IRC}_4)\\&\text{Explicit–Lagrange–Zarantonello}\\&\quad\text{with at least 7 implicit sequential stages } (\mathrm{ELZ}_7).\end{aligned}\tag{2.19}$$

Since the global order of PC methods equals $\min\{p, q\}$, it follows from Table 1 and Theorem 2.1 that the global orders of the methods ELC_2, IRC_4 and ELZ_7 are given by $\min\{p, 2m\}$ (recall that q^* equals the number of iterations), so that both the order and the stage order of the corrector is already reached for $m \geqslant p/2$. Hence, by satisfying the stability condition $m \geqslant m_0$, we are sure that the PC method has the same order p and stage order r as the corrector.

For completeness, we give the correctors selected above, and the corresponding vector β (see (2.4b)). Since these correctors originate from stiffly accurate RK methods, they all have $\alpha = e_k^T$.

The indirect Lagrange corrector with $k = 2$ is defined by

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{3}{4} & \frac{17}{128} & \frac{5}{16} & -\frac{21}{128} \\ 1 & \frac{11}{54} & \frac{14}{27} & -\frac{2}{9} \\ \hline & \frac{11}{54} & \frac{14}{27} & -\frac{2}{9} \\ & \frac{5}{18} & \frac{8}{9} & -\frac{1}{6} \end{array}, \qquad \beta = \left(-\tfrac{64}{9}, 6\right)^T.$$

The indirect Radau IIA corrector with $k = s - 1 = 3$ (written in the form (1.3′)) reads

$$\begin{array}{l|lll} 0.155051025722 & 0.021835034191 & -0.019857254099 & 0.010042630197 \\ 0.644948974278 & 0.177190587432 & 0.038164965809 & -0.007375963530 \\ 1.000000000000 & 0.318041381744 & 0.181958618256 & 0.000000000000 \\ \hline & 0.318041381744 & 0.181958618256 & 0.000000000000 \\ & 0.376403062700 & 0.512485826188 & 0.111111111111 \end{array},$$

with $\beta = (5.531972647422, -7.531972647422, 5)^T$.

The indirect Lagrange corrector with $k = 4$ is given by

0.0	0.0	0.0	0.0	0.0	0.0
0.166666666667	0.007240660574	0.008214814815	−0.003273544974	0.004739057239	−0.003032098765
0.583333333333	0.027437044052	0.114192181070	0.040901388889	−0.030997357838	0.018605632716
0.916666666667	0.034578097443	0.229227777778	0.165230621693	−0.032210648148	0.023313040123
1.000000000000	0.037142857143	0.256177777778	0.202057142857	−0.017777777778	0.022400000000
	0.037142857143	0.256177777778	0.202057142857	−0.017777777778	0.022400000000
	0.029870129870	0.325333333333	0.438857142857	0.193939393939	0.012000000000

with $\beta = (-4.8, 3.526530612245, -19.834710743802, 17.6)^T$.

3. Numerical comparisons

In this section, we restrict our considerations to the methods (2.19) and the two-, three- and four-stage SDIRKN methods of Section 1.2. In the experiments reported below, we dropped the fixed-number-of-iterations strategy used in the preceding examples. Instead, the number of iterations m was determined dynamically by the stability criterion $m \geqslant m_0$ together with a condition on the iteration error. It seems natural to require that the iteration error is of the same (stiff) order in h as the local error of the corrector. This leads us to the convergence criterion

$$m \geqslant m_0, \qquad \operatorname{Max}_i \| X_i^{(m)} - h^2 \sum_{j=1}^{k} a_{ij} g\left(X_j^{(m)} + x_j\right) \|_\infty \leqslant C h^{p+1}, \tag{3.1}$$

Table 8
Values of NCD/M for problem (2.15)

Method	k	p	r	$N=5$	$N=10$	$N=20$	$N=40$	$N=80$
Nørsett_2	2	3	1	0.9/10	1.8/20	2.7/40	3.6/80	4.5/160
SFB_2	2	3	1	0.6/10	1.5/20	2.4/40	3.3/80	4.2/160
Nørsett_3	3	4	1	3.1/15	3.1/30	4.1/60	5.2/120	6.4/240
SFB_3	3	4	1	2.4/15	3.6/30	4.8/60	6.0/120	7.2/240
B_4	4	3	1	0.9/20	1.8/40	2.7/80	3.6/160	4.5/320
ELC_2	2	3	3	2.0/19	2.9/39	3.8/78	4.6/156	
IRC_4	3	5	3	4.2/20	6.1/40	7.7/80	9.4/160	
ELZ_7	4	5	5	6.6/35	8.1/70	9.6/140	10.9/280	

where C is a parameter independent of h. In our numerical experiments we always used $C = 10^{-2}$.

Furthermore, in the tables of results, M denotes the (averaged) number of sequential systems to be solved per unit interval and N denotes the number of integration steps per unit interval.

3.1. Kramarz problem

Table 8 compares the methods specified above when applied to problem (2.15) of Kramarz. For this linear problem, where the Jacobian and its LU-decomposition can be computed once and for all at the beginning of the integration interval, the value of M may serve as a measure of the sequential computational costs. The results clearly show that the parallel methods IRC_4 and ELZ_7 are by far the most efficient ones, in spite of the fact that in this example no order reduction is observed. However, by the same reason, the method ELC_2 is only slightly more efficient than the other third-order methods. We observe that IRC_4 and ELZ_7 do not need more iterations to satisfy the convergence criterion (3.1) than already prescribed by the stability condition $m \geqslant m_0$.

3.2. Nonlinear partial differential equation

We apply the methods to the semidiscretization of the partial differential equation (see also [9])

$$\frac{\partial^2 u}{\partial t^2} = \frac{4\pi^2 u^2}{1 + 2x - 2x^2}\frac{\partial^2 u}{\partial x^2} + u\left[4\cos^2(2\pi t) - 1\right], \qquad 0 \leqslant t \leqslant 1, \quad 0 \leqslant x \leqslant 1, \tag{3.2}$$

with initial and Dirichlet boundary conditions such that its exact solution is given by $u = (1 + 2x - 2x^2)\cos(2\pi t)$. By using second-order symmetric spatial discretization on a uniform grid with mesh $\Delta x = \frac{1}{20}$, we obtain a set of 19 ODEs. Observe that this spatial discretization yields exact results for $\partial^2 u/\partial x^2$; hence, the exact solution of the system of ODEs is identical to the PDE solution, restricted to the gridpoints. Table 9 is the analogue of Table 8. Again, no order reduction is shown. If M is taken as a measure for the sequential computational costs, then only the four-processor method ELZ_7 can beat the one-processor methods. However, in this

Table 9
Values of NCD/M for problem (3.2)

Method	k	p	r	$N=40$	$N=80$	$N=160$	$N=320$
Nørsett_2	2	3	1	2.5/80	3.2/160	4.1/320	4.9/640
SFB_2	2	3	1	*	3.3/160	4.2/320	5.1/640
Nørsett_3	3	4	1	*	3.6/240	4.5/480	5.3/960
SFB_3	3	4	1	4.4/120	5.6/240	6.8/480	7.9/960
B_4	4	3	1	*	3.9/320	5.1/640	6.3/1280
ELC_2	2	3	3	3.8/296	4.7/566	5.5/946	6.4/1628
IRC_4	3	5	3	6.4/698	7.8/1126	9.2/1572	10.5/2274
ELZ_7	4	5	5	6.7/422	8.2/584	11.2/1120	13.1/2240

case of a semidiscrete nonlinear PDE, it is more realistic to consider the evaluations of the Jacobian and the corresponding LU-decompositions as the bulk of the computational work. This implies that all methods require approximately the same effort per step. As a consequence, both IRC_4 and ELZ_7 are the most efficient methods, while ELC_2 is only superseded by SFB_3. Notice that the residual condition in (3.1) now plays a dominant role in the determination of the number of iterations needed by the PC methods.

3.3. Prothero–Robinson-type problem

Consider the system of (uncoupled) second-order Prothero–Robinson-type equations (cf. [15]):

$$y''(t) = J[y(t) - g(t)] + g''(t),$$
$$J := \operatorname{diag}(-100^{j-1}), \qquad g(t) = (1 + e^{-jt}), \qquad j = 1, \ldots, 6; \quad 0 \leqslant t \leqslant 10, \tag{3.3}$$

with initial values $y(0) = g(0)$, $y'(0) = g'(0)$, so that its exact solution is given by $y(t) = g(t)$. For this problem, most methods show an irregular order behaviour, which is far from their theoretical (order p) behaviour. Hence, in this example, order reduction really occurs, which is caused by the stiffness of the problem. The results in Table 10 demonstrate the superiority of the methods IRC_4 and ELZ_7. The number of iterations in the PC methods is completely determined by the residual condition in (3.1).

Table 10
Values of NCD/M for problem (3.3)

Method	k	p	r	$N=1$	$N=2$	$N=4$	$N=8$	$N=16$
Nørsett_2	2	3	1	1.5/2	2.3/4	3.1/8	3.9/16	4.0/32
SFB_2	2	3	1	1.3/2	2.0/4	2.9/8	3.8/16	4.9/32
Nørsett_3	3	4	1	1.7/3	2.3/6	3.3/12	4.5/24	5.0/48
SFB_3	3	4	1	1.2/3	2.8/6	3.3/12	3.5/24	4.6/48
B_4	4	3	1	1.6/4	2.3/8	3.0/16	3.9/32	4.8/64
ELC_2	2	3	3	2.3/4	3.1/8	4.2/17	4.2/34	5.7/77
IRC_4	3	5	3	3.7/5	5.1/13	6.0/30	6.5/84	
ELZ_7	4	5	5	5.1/10	5.3/18	7.6/33	9.5/97	

Table 11
Values of NCD/M for problem (3.4) with M and N rounded to integer values

Method	k	p	r	$N=10$	$N=20$	$N=39$	$N=78$	$N=156$
Nørsett_2	2	3	1	0.1/20	0.1/39	0.6/78	1.5/157	2.4/313
SFB_2	2	3	1	0.1/20	0.1/39	0.4/78	1.2/157	2.1/313
Nørsett_3	3	4	1	0.1/29	0.2/59	0.8/117	1.8/235	3.1/470
SFB_3	3	4	1	−0.1/29	0.4/59	1.6/117	2.7/235	3.9/470
B_4	4	3	1	0.1/39	0.1/78	0.6/157	1.5/313	2.4/627
ELC_2	2	3	3	0.2/86	0.8/135	1.7/229	2.6/413	3.6/766
IRC_4	3	5	3	1.2/130	2.7/173	4.2/274	5.7/477	7.2/867
ELZ_7	4	5	5	2.5/75	4.1/137	5.7/274	7.2/548	8.7/1096

3.4. *Fehlberg problem*

Consider the nonlinear orbit equation (cf. [3]):

$$y''(t) = Jy(t), \qquad J := \begin{pmatrix} -4t^2 & -2/r(t) \\ 2/r(t) & -4t^2 \end{pmatrix}, \quad r(t) := \| y(t) \|_2; \quad \sqrt{\pi/2} \leqslant t \leqslant 3\pi,$$

with exact solution $y(t) = (\cos(t^2), \sin(t^2))^{\mathrm{T}}$. Similar to the previous example, we observe the order reduction phenomenon. As in the preceding examples, the methods IRC_4 and ELZ_7 are considerably more efficient, see Table 11.

4. Concluding remarks

Our starting point for the integration of systems of special second-order ODEs $y''(t) = f(y(t))$ with large Lipschitz constants is an *implicit* Runge–Kutta–Nyström method. Since a direct approach to solve the resulting system of nonlinear equations is not feasible because of its huge dimension (i.e., a multiple of the ODE dimension), we propose an *iterative* solution procedure.

To increase the efficiency, this iteration process is designed in such a way that it can be easily mapped onto a parallel computer architecture. This property is achieved by a so-called diagonal-implicit iteration which has the effect that—on each processor—a number of implicit relations has to be solved of a much lower dimension (i.e., the ODE dimension). Furthermore, the process has the additional advantage that (per processor) only one LU factorization per step is required.

The nature of the resulting algorithm is quite similar to so-called singly diagonally implicit RKN methods, several of which have been proposed in the literature or can easily be obtained from a singly diagonally implicit RK method for first-order ODEs. However, these schemes all suffer from the order reduction phenomenon due to their low stage order. This means that the observed order of convergence is much less than the classical order. Since our methods are based on a fully implicit RKN method which can easily be given a high stage order, the prospects for achieving an efficient behaviour are much better.

The technical part of the paper (Section 2) deals with the analysis of the iteration scheme and several approaches for choosing the free parameters in this iteration are discussed. On the

basis of its convergence analysis and the stability of the resulting method, we end up with an optimal selection consisting of (i) the underlying implicit RKN method, (ii) the iteration parameters and (iii) the predictor to start up the iteration. These specifications are given for methods to be implemented on parallel computers possessing 2, 3 or 4 (groups of) processors.

By means of four numerical examples it is shown that the new methods are much more efficient than the existing methods from the literature. This is due to the fact that they effectively exploit the parallel features of modern computers but also because they have a much higher (stage) order.

Since the successive iterations yield approximations of increasing order, a reference solution is available without additional costs, which can be used to extend the methods with error control and a varying stepsize strategy. Finally, the techniques described in this paper can straightforwardly be used to construct similar methods for the special ODE $y^{(\nu)}(t) = f(y(t))$, $\nu > 2$.

References

[1] K. Burrage, A study of order reduction for semi-linear problems, Report, University of Auckland (1990).

[2] K. Dekker and J.G. Verwer, *Stability of Runge–Kutta Methods for Stiff Nonlinear Differential Equations* (North-Holland, Amsterdam, 1984).

[3] E. Fehlberg, Klassische Runge–Kutta–Nyström Formeln mit Schrittweiten-Kontrolle für Differential-gleichungen $x'' = f(t, x)$, *Computing* 10 (1972) 305–315.

[4] E. Hairer, Unconditionally stable methods for second order differential equations, *Numer. Math.* 32 (1979) 373–379.

[5] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations* I, *Nonstiff Problems* (Springer, Berlin, 1987).

[6] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, *Stiff and Differential-Algebraic Systems* (Springer, Berlin, 1991).

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

[8] P.J. van der Houwen and B.P. Sommeijer, Iterated Runge–Kutta methods on parallel computers, *SIAM J. Sci. Statist. Comput.* 12 (1991) 1000–1028.

[9] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Stability of collocation-based Runge–Kutta–Nyström methods, *BIT* 31 (1991) 469–481.

[10] L. Kramarz, Stability of collocation methods for the numerical solution of $y'' = f(x, y)$, *BIT* 20 (1980) 215–222.

[11] I. Lie, Some aspects of parallel Runge–Kutta methods, Report No. 3/87, Division Numerical Mathematics, University of Trondheim (1987).

[12] T.A. Manteuffel, The Tchebyshev iteration for nonsymmetric linear systems, *Numer. Math.* 28 (1977) 307–327.

[13] S.P. Nørsett, Semi-explicit Runge–Kutta methods, Report Mathematics and Computation No. 6/74, Department of Mathematics, University of Trondheim (1974).

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

[15] A. Prothero and A. Robinson, On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations, *Math. Comp.* 28 (1974) 145–162.

[16] L.F. Shampine, Implementation of implicit formulas for the solution of ODEs, *SIAM J. Sci. Statist. Comput.* 1 (1980) 103–118.

[17] P.W. Sharp, J.H. Fine and K. Burrage, Two-stage and three-stage diagonally implicit Runge–Kutta–Nyström methods of orders three and four, *IMA J. Numer. Anal.* 10 (1990) 489–504.

[18] R.S. Varga, A comparison of the successive overrelaxation method and semi-iterative methods using Chebyshev polynomials, *SIAM J. Appl. Math.* 5 (1957) 39–47.

CHAPTER III

A-stable diagonally implicit Runge-Kutta-Nyström methods for parallel computers

published in: *Numerical Algorithms* **4** (1993), 263-281

A-stable diagonally implicit Runge–Kutta–Nyström methods for parallel computers

Nguyen huu Cong *

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands and Faculty of Mathematics, Mechanics and Informatics, University of Hanoi, Thuong dinh, Dong Da, Hanoi, Vietnam

Communicated by C. Brezinski
Received 21 April 1992; revised 12 November 1992

In this paper, we study diagonally implicit Runge–Kutta–Nyström methods (DIRKN methods) for use on parallel computers. These methods are obtained by diagonally implicit iteration of fully implicit Runge–Kutta–Nyström methods (corrector methods). The number of iterations is chosen such that the method has the same order of accuracy as the corrector, and the iteration parameters serve to make the method at least A-stable. Since a large number of the stages can be computed in parallel, the methods are very efficient on parallel computers. We derive a number of A-stable, strongly A-stable and L-stable DIRKN methods of order p with $s^*(p)$ sequential, singly diagonal-implicit stages where $s^*(p) = [(p+1)/2]$ or $s^*(p) = [(p+1)/2] + 1$, $[\cdot]$ denoting the integer part function.

Keywords: Diagonally implicit Runge–Kutta–Nyström methods, predictor–corrector methods, parallelism.

Subject classification: 65M12, 65M20.

1. Introduction

Consider the initial-value problem for systems of special second-order, ordinary differential equations (ODEs) of dimension d

$$y''(t) = f(y(t)), \quad y(t_0) = y_0, \quad y'(t_0) = y'_0,$$

$$y : \mathbb{R} \to \mathbb{R}^d, \quad f : \mathbb{R}^d \to \mathbb{R}^d, \quad t_0 \leqslant t \leqslant t_{\text{end}}. \tag{1.1}$$

One possibility for solving such problems is the use of singly diagonal-implicit Runge–Kutta–Nyström methods (SDIRKN methods). Compared with linear mul-

* These investigations were supported by the University of Amsterdam with a research grant to enable the author to spend a total of two years in Amsterdam.

© J.C. Baltzer AG Science Publishers

tistep methods (LM methods), SDIRKN methods have the disadvantage of requiring the solution of a sequence of implicit systems of dimension d per step, whereas LM methods require the solution of only one such system per step. On the other hand, a number of SDIRKN methods available in the literature possess excellent stability properties (cf. [17]), which are much better than those of the LM methods derived from the backward differentiation methods for first-order ODEs. In spite of that, LM methods are still more popular than SDIRKN methods, because of their lower costs on a sequential computer. However, on parallel computers, this situation may change. In this paper, we shall construct DIRKN methods tuned to parallel computers, such that each processor has to compute relatively few stages sequentially. We require that on each processor, these stages are *singly diagonal-implicit*, so that effectively the sequential costs of the parallel DIRKN method (PDIRKN method) are equal to those of an SDIRKN method. In fact, these methods are based on a *fixed* number of iterations of k-stage *indirect* RKN methods of Radau IIA and Gauss–Legendre type (methods of *indirect* type are understood to be methods that are derived by applying an RK method for first-order ODEs to the first-order form of (1.1)). Furthermore, the iteration parameters are chosen such that A-stability is obtained as soon as the order of the corrector is reached. The resulting methods require $k = [(p+1)/2]$ processors, where p denotes the order and $[\cdot]$ denotes the integer part function. We present a number of A-stable, strongly A-

Table 1
DIRKN methods of order p requiring s^* singly diagonal-implicit, sequential stages on k processors.

Method	p	s^*	k	Main properties	Type
Nørsett [15]	3	$p-1$	1	A-stable	indirect
Crouzeix [6]	3	$p-1$	1	Strongly A-stable	indirect
Sharp et al. [17]	3	$p-1$	1	A-stable, reduced phaselag	direct
Cash [3], Cash and Liem [4]	3	p	1	S-stable	indirect
Burrage [1]	3	$p+1$	1	A-stable, B-convergent	direct
Nørsett and Thomsen [16]	3	$p+1$	1	L-stable	indirect
Iserles and Nørsett [12]	4	$p-2$	2	L-stable	indirect
Nørsett [15]	4	$p-1$	1	A-stable	indirect
Sharp et al. [17]	4	$p-1$	1	A-stable, reduced phaselag	direct
Cash [3], Cash and Liem [4]	4	$p+1$	1	S-stable	indirect
Cooper and Sayfy [5]	5	p	1	A-stable	indirect
Van der Houwen et al. [9]	5	p	3	L-stable	indirect
Cooper and Sayfy [5]	6	$p-1$	1	A-stable	indirect
Sommeijer [18]	6	$p-1$	3	A-stable	indirect
Van der Houwen et al. [9]	6	p	3	L-stable	indirect
Van der Houwen et al. [9]	7	$p+1$	4	L-stable	indirect
	8	p	4	L-stable	indirect

stable and L-stable PDIRKN methods of order p with $s^*(p)$ sequential, singly diagonal-implicit stages, where $s^*(p) = [(p+1)/2]$ or $s^*(p) = [(p+1)/2] + 1$.

In order to appreciate these methods, we have summarized in table 1 the characteristics of a number of SDIRKN-type methods of orders $p = 3$ until $p = 8$. We included DIRKN methods of both *direct* and *indirect* type (for a specification of indirect RKN methods we refer to [10] and to the appendix of [14]). Furthermore, we also listed a few indirect parallel DIRKN methods derived from *parallel* DIRK methods. Both the sequential and parallel methods are (effectively) *singly* diagonal- implicit, so that the number of sequential stages s^* refers to the number of *singly* diagonal-implicit stages to be computed on each of the k processors.

By means of numerical experiments we will compare the performance of the methods constructed in this paper with that of a number of the methods listed in table 1.

2. Diagonal-implicit iteration

Our starting point is a fully implicit Runge–Kutta–Nyström (RKN) method of the form

$$y_{n+1} = y_n + hy'_n + h^2 \sum_{i=1}^{k} b_i f(Y_i),$$

$$y'_{n+1} = y'_n + h \sum_{i=1}^{k} d_i f(Y_i),$$

$$Y_i = y_n + c_i h y'_n + h^2 \sum_{j=1}^{k} a_{ij} f(Y_j), \quad i = 1, \ldots, k, \tag{2.1a}$$

where $\boldsymbol{b} = (b_i), \boldsymbol{c} = (c_i)$ and $\boldsymbol{d} = (d_i)$ are k-dimensional vectors, and $A = (a_{ij})$ is a nonsingular k-by-k matrix. This method will be referred to as the corrector method.

We employ a similar iteration technique as applied in [11] which automatically leads to DIRKN methods. Let $Y_i^{(\mu)}$ denote the μth iterate to Y_i, and define the transformed stage vector quantities X_i and $X_i^{(\mu)}$

$$X_i := Y_i - x_i, \quad X_i^{(\mu)} := Y_i^{(\mu)} - x_i, \quad x_i := y_n + c_i h y'_n, \quad i = 1, \ldots, k. \tag{2.1b}$$

These new variables are introduced in order to reduce round-off errors (cf. [8,p.128]). In terms of X_i and x_i, the stage vector equation in (2.1a) reads

$$X_i = h^2 \sum_{j=1}^{k} a_{ij} f(X_j + x_j), \quad i = 1, \ldots, k. \tag{2.1'a}$$

For each of these equations, we define the iteration process

$$X_i^{(\mu)} - \delta_i h^2 f(X_i^{(\mu)} + x_i) = h^2 \left(\sum_{j=1}^{k} a_{ij} f(X_j^{(\mu-1)} + x_j) - \delta_i f(X_i^{(\mu-1)} + x_i) \right), \quad (2.2a)$$

where $i = 1, \ldots, k; \mu = 1, \ldots, m$, the δ_i are positive iteration parameters, and where the initial approximations $X_i^{(0)}$ are to be provided by means of a predictor formula.

In this paper, we shall try to determine the iteration parameters such that the method is A-stable, strongly A-stable or L-stable as soon as the order of the corrector is reached. As we will see in sections 3 and 4, this can be achieved for a number of correctors derived from classical collocation correctors for first-order equations (indirect collocation correctors, specified in the appendix of the institute report [14]) using one-step predictor formulas of the form

$$X_i^{(0)} = \theta \delta_i h^2 f(X_i^{(0)} + x_i), \quad i = 1, \ldots, k, \quad (2.2b)$$

where either $\theta = 0$ or $\theta = 1$. These formulas will be referred to as predictor formulas of type I and II, respectively. The type I predictor $Y_i^{(0)} = x_i = y_n + c_i h y_n'$ is the trivial "last step value" predictor, which does not introduce amplification of stiff error components and does not require any additional computational effort. The type II predictor $Y_i^{(0)} = y_n + c_i h y_n' + \delta_i h^2 f(Y_i^{(0)})$ is implicit and may be considered as a "backward Euler type" predictor. Its strong stability properties may have a stabilizing effect on the whole method (strong damping of stiff components). For example, in the case of Radau correctors, it is possible to achieve L-stability by using type II predictors (see section 4). However, the price to be paid is an additional system of k implicit equations, the computational costs of which may be computed as an additional iteration (notice that the predictor formula of type II can use the same LU decomposition as needed in the subsequent iterations). Both types of predictors are first-order accurate. Within the class of one-step predictors, it is possible to achieve second-order accuracy. For example, we may define the explicit predictor

$$X_i^{(0)} = h^2 \sum_{j=1}^{k} a_{ij} f(y_n), \quad i = 1, \ldots, k.$$

However, such predictor formulas give rise to amplification of stiff components and is not suitable for our purposes. Since we preferred to stay within the class of one-step predictor–corrector methods, we did not investigate multistep predictors.

In [11] it was shown that the formulas for the step values defined in the corrector (2.1) can be presented in the form

$$y_{n+1} = y_n + h y_n' + \sum_{i=1}^{k} \alpha_i X_i, \quad y_{n+1}' = y_n' + h^{-1} \sum_{i=1}^{k} \beta_i X_i,$$

body0›where α_i and β_i are the components of the vectors $\boldsymbol{\alpha} := \boldsymbol{b}^{\mathrm{T}}A^{-1}, \boldsymbol{\beta} := \boldsymbol{d}^{\mathrm{T}}A^{-1}$. This suggests defining the step values y_{n+1} and y'_{n+1} corresponding to the iterated method as

$$y_{n+1} = y_n + hy'_n + \sum_{i=1}^{k} \alpha_i X_i^{(m)}, \quad y'_{n+1} = y'_n + h^{-1} \sum_{i=1}^{k} \beta_i X_i^{(m)}. \tag{2.3}$$

Since $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are not available in the literature, we have listed these vectors for the indirect collocation RKN correctors to be used in our numerical experiments (table 2). For stiffly accurate RKN correctors as Radau IIA, $\boldsymbol{\alpha} = \boldsymbol{e}_k^{\mathrm{T}}$.

We remark that for m fixed the method {(2.2), (2.3)} fits into the class of DIRKN methods that can be characterized by the Butcher array

$$\begin{array}{c|cccccc} X^{(0)} & \theta D & & & & & \\ X^{(1)} & A-D & D & & & & \\ X^{(2)} & O & A-D & D & & & \\ \vdots & & \ddots & \ddots & \ddots & & \\ X^{(m)} & O & O & O & \cdots\ O & A-D & D \\ \hline & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \cdots\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{b}^{\mathrm{T}}A^{-1}(A-D) & \boldsymbol{b}^{\mathrm{T}}A^{-1}D \\ & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \cdots\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{d}^{\mathrm{T}}A^{-1}(A-D) & \boldsymbol{d}^{\mathrm{T}}A^{-1}D \end{array} \tag{2.4}$$

where D is the diagonal matrix with diagonal entries δ_i. However, in an actual implementation, we shall use the representation {(2.2), (2.3)} which avoids f-evaluations in the step point formula.

Table 2
Vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for various indirect collocation RKN correctors.

Correctors	p	$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
Radau IIA	3	$\boldsymbol{\beta} = (-9/2, 5/2)^{\mathrm{T}}$
Gauss–Legendre	4	$\boldsymbol{\alpha} = (-1.732050807569, 1.732050807569)^{\mathrm{T}}$
		$\boldsymbol{\beta} = (-16.392304845413, 4.392304845413)^{\mathrm{T}}$
Radau IIA	5	$\boldsymbol{\beta} = (5.531972647422, -7.531972647422, 5)^{\mathrm{T}}$
Gauss–Legendre	6	$\boldsymbol{\alpha} = (5/3, -4/3, 5/3)^{\mathrm{T}}$
		$\boldsymbol{\beta} = (32.909944487358, -16, 7.090055512642)^{\mathrm{T}}$
Radau IIA	7	$\boldsymbol{\beta} = (-6.923488256444, 6.595237669626, -12.171749413180, 17/2)^{\mathrm{T}}$
Gauss–Legendre	8	$\boldsymbol{\alpha} = (-1.640705321739, 1.214393969799, -1.214393969799, 1.640705321739)^{\mathrm{T}}$
		$\boldsymbol{\beta} = (-54.681428514064, 26.155201475250, -22.420557316693, 10.946784355507)^{\mathrm{T}}$

Since the k systems that are to be solved in each iteration step of (2.2) can be solved *in parallel* and each has a dimension equal to that of the system of ODEs, the iteration process (2.2) is, on a k-processor computer, of the same computational complexity as an $(m+\theta)$-stage SDIRKN method on a one-processor computer. Thus, the method {(2.2), (2.3)} has only $s^* := m + \theta$ *sequential, singly diagonal-implicit* stages.

THEOREM 2.1

Let p be the order of the k-stage corrector method (2.1) and let $m := [(p+1)/2]$. Then the method {(2.2), (2.3)} is an s-stage DIRKN method of order p with s^* sequential, singly diagonal-implicit stages, where s and s^* are defined by $s = k[(p+1)/2] + 1 + \theta(k-1)$ and $s^* = [(p+1)/2] + \theta$.

Proof

The expressions for s and s^* immediately follow from the Butcher array (2.4). The order of the method is obtained by considering the iteration error of the method. Since (2.2b) defines a first-order predictor formula, we have $X_i^{(0)} - X_i = O(h^2)$. Furthermore, subtracting (2.1'a) and (2.2a) yields

$$X_i^{(\mu)} - X_i - \delta_i h^2 (f(X_i^{(\mu)} + x_i) - f(X_i + x_i)) = -\delta_i h^2 (f(X_i^{(\mu-1)} + x_i) - f(X_i + x_i)) + h^2 \sum_{j=1}^{k} a_{ij} (f(X_j^{(\mu-1)} + x_j) - f(X_j + x_j)) .$$

Assuming that f has a bounded Lipschitz constant, it follows that $X_i^{(\mu)} - X_i = O(h^2)(X_i^{(\mu-1)} - X_i)$, so that

$$X_i^{(m)} - X_i = O(h^{2+2m}) . \tag{2.5}$$

In order to avoid confusion, let us denote the step values associated with the corrector by u_{n+1} and u'_{n+1}. Subtracting the corrector step values and the iterated step values shows that

$$u_{n+1} - y_{n+1} = \sum_{i=1}^{k} \alpha_i (X_i - X_i^{(m)}) = O(h^{2+2m}),$$

$$u'_{n+1} - y'_{n+1} = h^{-1} \sum_{i=1}^{k} \beta_i (X_i - X_i^{(m)}) = O(h^{1+2m}).$$

Let $y(t)$ be the local exact solution. Then the local truncation error is given by

$$\begin{aligned} y(t_{n+1}) - y_{n+1} &= y(t_{n+1}) - u_{n+1} + u_{n+1} - y_{n+1} = O(h^{p+1}) + O(h^{2+2m}) , \\ y'(t_{n+1}) - y'_{n+1} &= y'(t_{n+1}) - u'_{n+1} + u'_{n+1} - y'_{n+1} = O(h^{p+1}) + O(h^{1+2m}) , \end{aligned} \tag{2.6}$$

where p is the order of the corrector. Thus, we need only $m = [(p+1)/2]$ iterations to reach the order of the corrector, so that $s^* := m + \theta = [(p+1)/2] + \theta$. □

It follows from (2.6) that there are three sources of local errors which together constitute the global error, i.e., the truncation error of the corrector (of order $p+1$) and the iteration errors corresponding to y_{n+1} and y'_{n+1} (of orders $2m+2$ and $2m+1$). In addition to these orders, the order *constants* also play a role. The magnitude of the order constant associated with the corrector is usually rather small. The order constants of the iteration errors decrease with m and are expected to be rather large for small values of m (see also table 3). As the value of m is relatively small, the iteration errors may easily dominate the global error, so that the order of the corrector is not always shown in actual computation. For example, if the iteration error corresponding to y_{n+1} dominates, then the effective order p^* is given by $p^* = 2m+1 = 2[(p+1)/2]+1$. Likewise, if the iteration error corresponding to y'_{n+1} dominates, then $p^* = 2m = 2[(p+1)/2]$. However, if the integration stepsize h is sufficiently small, then the iteration errors should become negligible, so that the truncation error of the corrector method dominates, and the theoretical order of the corrector should be shown (see table 7).

3. Stability

The linear stability of the method {(2.2), (2.3)} is determined by applying it to the scalar test equation $y'' = \lambda y$, where λ runs through the eigenvalues of $\partial f/\partial y$, which are supposed to be negative. Defining the matrix

$$Z(z) := z[I - zD]^{-1}[A - D], \quad P_\theta(z) := z[I - zA]^{-1}[A - \theta D][I - \theta z D]^{-1},$$

$$z := \lambda h^2, \tag{3.1}$$

and the vectors

$$w_{n+1} := \begin{pmatrix} u_{n+1} \\ hu'_{n+1} \end{pmatrix}, \quad v_{n+1} := \begin{pmatrix} y_{n+1} \\ hy'_{n+1} \end{pmatrix}, \tag{3.2}$$

it can be shown (cf. [11]) that the following recursions hold:

$$w_{n+1} - v_{n+1} = E_m(z)v_n,$$

$$E_m(z) := \begin{pmatrix} b^{\mathrm{T}}A^{-1}Z^m(z)P_\theta(z)e & b^{\mathrm{T}}A^{-1}Z^m(z)P_\theta(z)c \\ d^{\mathrm{T}}A^{-1}Z^m(z)P_\theta(z)e & d^{\mathrm{T}}A^{-1}Z^m(z)P_\theta(z)c \end{pmatrix}, \tag{3.3}$$

$$w_{n+1} = M(z)v_n,$$

$$M(z) := \begin{pmatrix} 1 + zb^{\mathrm{T}}(I - Az)^{-1}e & 1 + zb^{\mathrm{T}}(I - Az)^{-1}c \\ zd^{\mathrm{T}}(I - Az)^{-1}e & 1 + zd^{\mathrm{T}}(I - Az)^{-1}c \end{pmatrix}. \tag{3.4}$$

Hence, by eliminating the corrector values w_{n+1} from (3.3) and (3.4), we find the recursion

$$v_{n+1} = [M(z) - E_m(z)]v_n . \tag{3.5}$$

We shall call the matrix $M(z) - E_m(z)$ the *stability matrix* of the method and its spectral radius the *stability function*, i.e., the function:

$$R_m(z) := \rho([M(z) - E_m(z)]).$$

The method {(2.2), (2.3)} is called A-stable if $R_m(z)$ assumes values in $(-1, 1)$ for $z < 0$, strongly A-stable if it is A-stable with $R_m(z)$ bounded away from 1 outside the neighbourhood of the origin, and L- stable if it is A-stable with $R_m(\infty) = 0$.

Putting $m = [(p + 1)/2]$, we obtain pth-order accuracy for any D. We shall exploit the matrix D to obtain pth-order A-stable, strongly A-stable or L-stable methods. However, it turns out that various choices of D generate such highly stable methods. From these methods we selected the methods with smallest truncation error. Recalling that the truncation error of the PDIRKN method will usually be dominated by the iteration error, we are led to consider the iteration error defined by (3.3). Since the nonstiff error components in the iteration error corresponding to small values of $|z|$ are sufficiently damped by the matrix $E_m(z)$ (note that $E_m(z) = O(z^{m+1})$), we shall concentrate on the *stiff* error components. From (3.2), (3.3) and (3.4) it follows that

$$\begin{aligned} w_{n+1} - v_{n+1} &= E_m(z)v_n \\ &= E_m(z)[M(z) - E_m(z)]v_{n-1} \\ &= E_m(z)[M(z) - E_m(z)]^n v_0. \end{aligned}$$

Restricting our considerations to the iteration error associated with y_{n+1}, we deduce that $u_{n+1} - y_{n+1}$ can be bounded by

$$\begin{aligned} \|u_{n+1} - y_{n+1}\| &= \|e_1^{\mathrm{T}} E_m(z)[M(z) - E_m(z)]^n v_0\| \\ &\leqslant \|e_1^{\mathrm{T}} E_m(z)\| \|[M(z) - E_m(z)]^n\| \|v_0\| \\ &\approx \text{const.} n^{\nu-1}[R_m(z)]^n \|e_1^{\mathrm{T}} E_m(z)\| \|v_0\| \quad \text{as } n \to \infty, \end{aligned} \tag{3.6}$$

where ν denotes the maximum dimension of the Jordan box corresponding to the maximum-modulus-eigenvalues of the matrix $M(z) - E_m(z)$. This estimate shows that the stiff error components can be suppressed if the stability function $R_m(z)$ is small for large $|z|$-values. We remark that a similar estimate can be derived for $u'_{n+1} - y'_{n+1}$. The following theorem may be helpful in selecting methods possessing this property:

THEOREM 3.1

Let the predictor be given by (2.2b) and let the corrector (2.1) be obtained from a consistent RK method for first-order equations given by the parameter arrays $\{A^*, b^*, c\}$, then the following assertions hold:

(a) If $\theta = 0$, then

$$R_m(\infty) = \rho\begin{pmatrix} 1 - (b^*)^{\mathrm{T}} A^* Q_m e & 1 - (b^*)^{\mathrm{T}} A^* Q_m c \\ -(b^*)^{\mathrm{T}} Q_m e & 1 - (b^*)^{\mathrm{T}} Q_m c \end{pmatrix},$$

$$Q_m := (A^*)^{-2}[I - [I - D^{-1}(A^*)^2]^m].$$

(b) If $\theta = 1$, then $R_m(\infty) = |1 - (b^*)^{\mathrm{T}}(A^*)^{-1}e|$ for all m and D, and if the RK method $\{A^*, b^*, c\}$ is stiffly accurate, then $R_m(\infty) = 0$ for all m and D.

Proof

If the corrector (2.1) is obtained from an RK method for first-order equations $\{A^*, b^*, c\}$, then

$$A = (A^*)^2, \quad b = (A^*)^{\mathrm{T}} b^*, \quad c = A^* e, \quad d = b^*. \tag{3.7}$$

Furthermore, we have that $Z(\infty) = I - D^{-1}A$ and $P_\theta(\infty) = (\theta - 1)I$, where θ is either 0 or 1. Hence,

$$M(\infty) - E_m(\infty) = \begin{pmatrix} 1 - b^{\mathrm{T}} Q_{m\theta} e & 1 - b^{\mathrm{T}} Q_{m\theta} c \\ -d^{\mathrm{T}} Q_{m\theta} e & 1 - d^{\mathrm{T}} Q_{m\theta} c \end{pmatrix},$$

$$Q_{m\theta} := A^{-1}[I + (\theta - 1)[I - D^{-1}A]^m]. \tag{3.8}$$

(a) On substitution of $\theta = 0$ and (3.7) into (3.8), part (a) is immediate.
(b) For $\theta = 1$ and using (3.7), we see that (3.8) reduces to

$$M(\infty) - E_m(\infty) = \begin{pmatrix} 1 - (b^*)^{\mathrm{T}}(A^*)^{-1}e & 1 - (b^*)^{\mathrm{T}} e \\ -(b^*)^{\mathrm{T}}(A^*)^{-2}e & 1 - (b^*)^{\mathrm{T}}(A^*)^{-1}e \end{pmatrix}. \tag{3.8'}$$

Because of the consistency we have that $(b^*)^{\mathrm{T}} e = 1$, so that the eigenvalues of $M(\infty) - E_m(\infty)$ are given by $1 - (b^*)^{\mathrm{T}}(A^*)^{-1}e$. If the corrector $\{A^*, b^*, c\}$ is stiffly accurate, then

$$e_k^{\mathrm{T}} c = 1, \quad (b^*)^{\mathrm{T}} = e_k^{\mathrm{T}} A^*, \tag{3.9}$$

so that $R_m(\infty)$ vanishes for all m and D.

This theorem shows that for *explicit* predictors of type I ($\theta = 0$), the behaviour of the stability function at infinity depends on D, so that we can exploit the matrix D by selecting methods with the smallest value $R_m(\infty)$. It is interesting to note that we obtained strongly A-stable PDIRKN methods although the corrector is only A-stable (e.g., in the case of Gauss–Legendre correctors listed in table 3).

For *implicit* predictors of type II ($\theta = 1$), the behaviour of the stability function at infinity is completely determined by the corrector, so that D cannot be used for selecting small values of $R_m(\infty)$ in the estimate (3.6). However, (3.6) indicates that the iteration error is also influenced by the magnitude of $\|e_1^{\mathrm{T}} E_m(z)\|$. Since

Table 3
PDIRKN methods of order p requiring s^* singly diagonal-implicit, sequential stages on k processors.

{Predictor–Corrector}	Iteration parameters δ_l	p	s^*	k	Stability	$E_{\max}$	E_∞
{I - Radau IIA}	(11/200, 107/225)	3	$p-1$	2	Strongly A-stable	0.35	0.06
{II - Radau IIA}	(1/5, 1/5)	3	p	2	L-stable	0.14	0.00
{I - Gauss-Legendre}	(1/5, 11/20)	4	$p-2$	2	Strongly A-stable	1.35	1.35
{II - Gauss-Legendre}	(223/10000, 311/1000)	4	$p-1$	2	A-stable	0.25	0.00
{I - Radau IIA}	(1/40, 1/4, 3/5)	5	$p-2$	3	Strongly A-stable	0.73	0.16
{II - Radau IIA}	(639/5000, 17/1250, 409/2500)	5	$p-1$	3	L-stable	0.51	0.00
{I - Gauss-Legendre}	(1/5, 1/2, 3/4)	6	$p-3$	3	Strongly A-stable	1.44	0.51
{II - Gauss-Legendre}	(1/100, 1/5, 9/20)	6	$p-2$	3	A-stable	1.32	0.00
{I - Radau IIA}	(1/5, 4/5, 4/5, 19/20)	7	$p-3$	4	Strongly A-stable	1.43	0.77
{II - Radau IIA}	(9/200, 1/40, 9/40, 91/200)	7	$p-2$	4	L-stable	1.09	0.00
{I - Gauss-Legendre}	(13/20, 13/20, 3/4, 19/20)	8	$p-4$	4	Strongly A-stable	1.60	1.60
{II - Gauss-Legendre}	(1/10, 1/5, 3/10, 2/5)	8	$p-3$	4	A-stable	1.55	0.00

$e_1^T E_m(z)$ vanishes at infinity, we selected methods with a small value of $\|e_1^T E_m(z)\|$ in the whole interval $(-\infty, 0)$.

Finally, we remark that the preceding discussion of the error $u_{n+1} - y_{n+1}$ can also be given for the derivative error $u'_{n+1} - y'_{n+1}$, presumably leading to other matrices D. As a consequence, the PDIRKN methods using the D matrices indicated above aim at problems where our first interest is in an accurate computation of the solution $y(t)$, rather than $y'(t)$.

4. Survey of PDIRKN methods

In table 3, we list the main characteristics of the A-stable, strongly A-stable and L-stable PDIRKN methods we found by means of the approach described in the preceding sections. In this table, $E_{\max}$ denotes the maximum value of $\|e_1^T E_m(z)\|_\infty$ in the interval $(-\infty, 0)$ and E_∞ denotes the value of $\|e_1^T E_m(\infty)\|_\infty$. The predictors are of the form (2.2b) with $\theta = 0$ (predictor I) and $\theta = 1$ (predictor II), and the correctors used are the indirect collocation-type RKN methods based on the Gauss–Legendre and Radau IIA RK methods for first-order equations. Specification of the parameters of the resulting methods can be found in the appendix to [14].

Comparing the main characteristics of the methods listed in table 3 with those listed in table 1, we conclude that the computational costs per step of the lower-order methods (order three or four) are comparable, but the higher-order methods in table 3 are much cheaper. On the other hand, the error constant $E_{\max}$ of the itera-

tion error associated with y_{n+1} is relatively large. However, as we have shown in the discussion of theorem 2.1, the order in h of these iteration errors is also larger, which may compensate the large error constants. Hence, we may hope for improved efficiency for the new PDIRKN methods.

5. Numerical experiments

We shall numerically investigate the following aspects of the PDIRKN methods: (i) the stability, in particular, the damping of perturbations of the initial conditions, (ii) the effective order, in relation to the order of the generating corrector, (iii) the predictor, mutual comparison of the explicit and implicit predictor formula, and (iv) the efficiency, in comparison with available sequential SDIRKN methods from the literature.

All problems are taken from the literature and possess exact solutions in closed form. Initial (and boundary) conditions are taken from the exact solution. Most experiments are performed on a 14 digit computer. Only the results reported in table 7 are performed in double precision (28 digits). Furthermore, because of round-off errors, we cannot expect 14 digits or 28 digits accuracy. As a consequence, the tables of results do contain empty spots whenever the corresponding numerical result was in the neighbourhood of the accuracy-limits of the machine and therefore considered as unreliable.

5.1. STABILITY TEST

We first test the stability properties of the various PDIRKN methods by integrating a nonautonomous problem with varying stiffness:

$$y''(t) = \begin{pmatrix} -2\alpha(t)+1 & -\alpha(t)+1 \\ 2(\alpha(t)-1) & \alpha(t)-2 \end{pmatrix} y(t),$$

$$y(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad y'(0) = \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \quad 0 \leqslant t \leqslant T,$$

$$\alpha(t) = \sqrt{1+t^3} + \frac{1}{\sqrt{1+t^3}}. \tag{5.1}$$

The Jacobian matrix of the system has the eigenvalues -1 and $-\alpha(t)$, so that the spectral radius, and therefore the stiffness, increases with t. We compared the numerical solution of (5.1) with the numerical solution obtained by perturbing the initial conditions, i.e., instead of the initial conditions $y(0)$ and $y'(0)$ we used the initial conditions $y(0) + \epsilon e$ and $y'(0) + \epsilon e$. Denoting the numerical solutions by y_n and y_n^*, we may expect from any stable method that $\|y_n - y_n^*\|$ does not increase with n. For various PDIRKN methods, table 4 lists the values

Table 4
Values of the amplification factor C_n for problem (5.1) with $T = 4000$, $n = 4000$ and with $T = 6000, n = 6000$ for various Predictor–Corrector pairs.

Type I Methods	p	C_{4000}	C_{6000}	Type II Methods	p	C_{4000}	C_{6000}
I - Radau IIA	3	0.30E-23	0.84E-36	II - Radau IIA	3	0.43E-11	0.35E-17
I - Gauss–Legendre	4	0.44E-12	0.22E-18	II - Gauss–Legendre	4	0.44E-01	0.38E-01
I - Radau IIA	5	0.53E-29	0.48E-44	II - Radau IIA	5	0.64E-01	0.67E-02
I - Gauss–Legendre	6	0.17E-23	0.12E-35	II - Gauss–Legendre	6	0.34E-13	0.93E-14
I - Radau IIA	7	0.88E-21	0.63E-31	II - Radau IIA	7	0.40E-09	0.61E-15
I - Gauss–Legendre	8	0.53E-25	0.13E-38	II - Gauss–Legendre	8	0.44E-12	0.51E-13

$$C_n := \|y_n - y_n^*\| / \|y_0 - y_0^*\|$$
$$= \|y_n - y_n^*\| / \epsilon \quad \text{for } n = 4000 \text{ and } n = 6000.$$

The methods are specified by the generating Predictor–Corrector pair where the predictor is indicated by its type. It turned out that C_n is almost independent of ϵ for $\epsilon \leqslant 1/10$. The results in table 4 demonstrate the strong damping of the initial perturbation by all PDIRKN methods.
We remark that with respect to the scalar test equation (see also (3.6)), the estimate

$$\|y_n - y_n^*\| = \|e_1^T (M_m(z) - E_m(z))^n (y_0 - y_0^*)\|$$
$$\approx \text{const.} n^{\nu-1} [R(z)]^n \epsilon$$

shows that C_n depends on the stability behaviour of the PDIRKN method for a particular value of z, and it is expected that for an A-stable PDIRKN method and a given problem with specified stepsize, C_n will decrease as n increases. This behaviour is demonstrated by the results listed in table 4.

Another observation is that for this linear problem, the explicit predictors give a better damping than the implicit predictors. The damping effect turns out to be strongly problem-dependent as is shown by the following example:

$$y''(t) = -1000(y(t) - \cos(t))^3 - \cos(t), \quad y(0) = 1, y'(0) = 0, 0 \leqslant t \leqslant T. \quad (5.1')$$

Applying the same test strategy as before, the results listed in table 5 show that the implicit and explicit predictors give rise to a similar damping effect for this problem. Moreover, the damping is much weaker when compared to the previous example.

5.2. EFFECTIVE ORDER AND EFFICIENCY OF THE EXPLICIT AND IMPLICIT PREDICTOR

In this section, we show that the effective order of the PDIRKN methods may exceed the order of the corrector. In addition, we compare the efficiency of the explicit and implicit predictor. In all experiments the accuracy is given by means of the number of minimal correct digits (NCD) defined by NCD(h) = $-\log(\|$ global

Table 5
Values of the amplification factor C_n for problem (5.1') with $T = 1000, n = 10000$ for various Predictor–Corrector pairs.

Type I Methods	p	s^*	k	C_n	Type II Methods	p	s^*	k	C_n
I - Radau IIA	3	2	2	0.36E+00	II - Radau IIA	3	3	2	0.63E+00
I - Gauss–Legendre	4	2	2	0.44E+00	II -Gauss–Legendre	4	3	2	0.12E+00
I - Radau IIA	5	3	3	0.82E+00	II - Radau IIA	5	4	3	0.58E+00
I - Gauss–Legendre	6	3	3	0.89E+00	II -Gauss–Legendre	6	4	3	0.10E+01
I - Radau IIA	7	4	4	0.10E+01	II - Radau IIA	7	5	4	0.64E+00
I - Gauss–Legendre	8	4	4	0.10E+01	II -Gauss–Legendre	8	5	4	0.48E+00

error at the endpoint of the integration interval $\|\|_\infty$), and the computational effort is measured by the number of sequential stages per unit interval. The (fixed) stepsize is chosen such that the number of sequential stages per unit interval (approximately) equals a prescribed number M. To be more precise, let N_{steps} denote the total number of integration steps for the integration interval $[t_0, T]$, then $M = N_{\text{steps}} s^*/(T - t_0)$, which leads us to

$$N_{\text{steps}} = \left[\frac{M(T - t_0)}{s^*} + 0.5\right], \quad h = \frac{T - t_0}{N_{\text{steps}}},$$

where $[\cdot]$ denotes the integer part function (the effect of the $[\cdot]$ operation causes that the actual number of sequential stages may be slightly different from the prescribed number M).

Table 6 lists results for the linear Kramarz problem (see [13])

$$y''(t) = \begin{pmatrix} 2498 & 4998 \\ -2499 & -4999 \end{pmatrix} y(t), \quad 0 \leqslant t \leqslant 100, \tag{5.2}$$

Table 6
Effective order p^* and values of NCD and M for problem (5.2).

Predictor–Corrector	p	s^*	k	$M = 25$	$M = 50$	$M = 100$	$M = 200$	p^*
I - Radau IIA	3	2	2	2.8	3.8	4.7	5.6	3
II - Radau IIA	3	3	2	2.4	3.3	4.2	5.1	3
I - Gauss–Legendre	4	2	2	3.3	4.5	5.7	6.9	4
II - Gauss–Legendre	4	3	2	4.0	5.4	6.7	8.0	4
I - Radau IIA	**5**	**3**	**3**	**4.2**	**6.0**	**7.8**	**9.6**	**6**
II - Radau IIA	5	4	3	5.1	6.8	8.5	10.0	5
I - Gauss–Legendre	6	3	3	3.9	5.8	7.6	9.4	6
II - Gauss–Legendre	**6**	**4**	**3**	**4.6**	**6.7**	**8.8**	**11.0**	**7**
I - Radau IIA	**7**	**4**	**4**	**4.5**	**6.9**	**9.3**	**12.0**	**8**
II - Radau IIA	**7**	**5**	**4**	**5.4**	**8.1**	**10.8**		**9**
I - Gauss–Legendre	8	4	4	4.4	6.8	9.2	12.8	8
II - Gauss–Legendre	8	5	4	5.2	7.7	10.1		8

Table 7
Effective order p^* and values of NCD and M for problem (5.2) obtained by some specified PDIRKN methods with small stepsizes.

Predictor–Corrector	p	s^*	k	$M = 800$	$M = 1600$	$M = 3200$	$M = 6400$	p^*
I - Radau IIA	5	3	3	13.1	14.8	16.5	18.1	5.3
II - Gauss–Legendre	6	4	3	15.3	17.3	19.2	21.0	6
I - Radau IIA	7	4	4	16.5	18.7	20.9		7.3
II - Radau IIA	7	5	4	18.5	20.7	22.7		6.7

with exact solution $y(t) = (2\cos(t), -\cos(t))^T$. These results show that for some higher-order methods (indicated in bold face), the measured effective order p^* is greater than p (see the discussion of theorem 2.1). In order to show that this "higher-order behaviour" is caused by a dominance of the iteration error, we applied these "higher-order" PDIRKN methods again to the Kramarz problem (5.2), but now with very small stepsizes. Using a high-precision computer (28 digits), we obtained the results listed in table 7, showing that the corrector-order is more or less retained.

Finally, we observe that usually the implicit predictor (type II) produces better results, in spite of the additional implicit stage. Therefore, in the following, we shall confine our considerations to the type II predictor.

5.3. EFFICIENCY TESTS

In this section, we compare the efficiency of the PDIRKN method with methods from the literature. We selected the following methods from table 1:

Table 8
Values of NCD and M for problem (5.2).

Methods	p	s^*	k	$M = 25$	$M = 50$	$M = 100$	$M = 200$
Nørsett_3	3	2	1	2.1	3.0	3.9	4.8
SFB_3	3	2	1	1.8	2.7	3.6	4.5
B_3	3	4	1	1.2	2.1	3.0	3.9
II - Radau IIA	3	3	2	2.4	3.3	4.2	5.1
Nørsett_4	4	3	1	2.8	3.8	4.9	6.1
SFB_4	4	3	1	3.2	4.5	5.7	6.9
II - Gauss–Legendre	4	3	2	4.0	5.4	6.7	8.0
CS_5	5	5	1	4.1	5.6	7.1	8.6
II - Radau IIA	5	4	3	5.1	6.8	8.5	10.0
CS_6	6	5	1	5.5	7.0	8.4	9.0
II - Gauss–Legendre	6	4	3	4.6	6.7	8.8	11.0
II - Radau IIA	7	5	4	5.4	8.1	10.8	
II - Gauss–Legendre	8	5	4	5.2	7.7	10.1	

Nørsett_3	third-order method of Nørsett;
Nørsett_4	fourth-order method of Nørsett;
SFB_3	third-order method of Sharp et al.;
SFB_4	fourth-order method of Sharp et al.;
B_3	third-order method of Burrage;
CS_5	fifth-order method of Cooper and Sayfy;
CS_6	sixth-order method of Cooper and Sayfy.

5.3.1. Linear Kramarz problem

Table 8 presents results for these sequential methods and for our PDIRKN methods when applied to the Kramarz problem (5.2). In most cases, the PDIRKN methods are by far the most accurate ones. Notice that the CS_6 method does not show its order 6 in the high accuracy range. This is caused by an insufficient accuracy of the method parameters. As a consequence, the CS_6 method may well be competitive with the sixth-order PDIRKN method.

5.3.2. Linear Strehmel–Weiner problem

In [19] we find the following linear, stiff problem:

$$y''(t) = \begin{pmatrix} -20.2 & 0 & -9.6 \\ 7989.6 & -10000 & -6004.2 \\ -9.6 & 0 & -5.8 \end{pmatrix} y(t) + \begin{pmatrix} 150\cos(10t) \\ 75\cos(10t) \\ 75\cos(10t) \end{pmatrix},$$

$$0 \leqslant t \leqslant 100, \tag{5.3}$$

Table 9
Values of NCD and M for problem (5.3).

Methods	p	s^*	k	$M = 100$	$M = 200$	$M = 400$	$M = 800$
Nørsett_3	3	2	1	1.1	2.0	2.9	3.8
SFB_3	3	2	1	0.8	1.7	2.6	3.5
B_3	3	4	1	0.3	1.1	2.0	2.9
II - Radau IIA	3	3	2	1.4	2.3	3.2	4.1
Nørsett_4	4	3	1	1.2	2.5	3.8	5.0
SFB_4	4	3	1	2.3	3.4	4.7	5.9
II - Gauss–Legendre	4	3	2	3.1	4.9	6.7	7.3
CS_5	5	5	1	3.0	4.5	5.9	7.4
II - Radau IIA	5	4	3	4.9	6.6	7.6	9.0
CS_6	6	5	1	3.6	5.5	7.5	8.2
II - Gauss–Legendre	6	4	3	3.2	5.3	7.4	9.4
II - Radau IIA	7	5	4	3.9	6.6	9.4	10.0
II - Gauss–Legendre	8	5	4	4.4	6.5	8.8	10.0

with exact solution

$$y(t) = \begin{pmatrix} \cos(t) + 2\cos(5t) - 2\cos(10t) \\ 2\cos(t) + \cos(5t) - \cos(10t) \\ -2\cos(t) + \cos(5t) - \cos(10t) \end{pmatrix}.$$

Unlike the Kramarz problem, this problem has slowly and rapidly oscillating solution components (nonstiff and stiff solution components) which are appearing with comparable weights. This implies a severe test for the PDIRKN methods because of the strong damping, and therefore inaccurate approximation, of the stiff solution components. In spite of that, they are generally superior to the sequential methods. Again, taking into account the inaccurate method parameters of CS_6, we see from the results listed in table 9 that this method is competitive.

5.3.3. Nonlinear Strehmel–Weiner problem

In [19] we also find a nonlinear, stiff problem:

$$y_1''(t) = (y_1(t) - y_2(t))^3 + 6368y_1(t) - 6384y_2(t) + 42\cos(10t),$$

$$y_2''(t) = -(y_1(t) - y_2(t))^3 + 12768y_1(t) - 12784y_2(t) + 42\cos(10t),$$

$$0 \leqslant t \leqslant 10, \tag{5.4}$$

with exact solution $y_1(t) = y_2(t) = \cos(4t) - \cos(10t)/2$. Table 10 demonstrates that the PDIRKN methods similarly compare with the sequential methods as for the linear Kramarz and Strehmel–Weiner problems.

Table 10
Values of NCD and M for problem (5.4).

Methods	p	s^*	k	$M = 100$	$M = 200$	$M = 400$	$M = 800$
$Nørsett_3$	3	2	1	2.9	3.9	4.8	5.7
SFB_3	3	2	1	2.7	3.6	4.5	5.4
B_3	3	4	1	2.3	3.6	5.2	6.2
II - Radau IIA	3	3	2	3.3	4.1	5.1	6.0
$Nørsett_4$	4	3	1	3.0	4.2	5.3	6.5
SFB_4	4	3	1	3.7	4.9	6.1	7.3
II - Gauss–Legendre	4	3	2	4.8	6.1	7.4	8.7
CS_5	5	5	1	4.9	6.4	7.9	9.4
II - Radau IIA	5	4	3	5.8	7.6	9.4	11.1
CS_6	6	5	1	5.9	7.6	9.2	9.9
II - Gauss–Legendre	6	4	3	5.5	7.6	9.7	11.8
II - Radau IIA	7	5	4	6.4	9.0	11.6	
II - Gauss–Legendre	8	5	4	5.8	8.2	10.6	

5.3.4. Fehlberg problem

An often used test problem is the orbit equation (cf. [7])

$$y_1''(t) = -4t^2 y_1(t) - \frac{2y_2(t)}{\sqrt{y_1^2(t) + y_2^2(t)}},$$
$$y_2''(t) = -4t^2 y_2(t) + \frac{2y_1(t)}{\sqrt{y_1^2(t) + y_2^2(t)}}, \qquad \sqrt{\pi/2} \leqslant t \leqslant 3\pi, \tag{5.5}$$

with the exact solution $y_1(t) = \cos(t^2), y_2(t) = \sin(t^2)$. Results are presented in table 11. Usually this type of equations has to be solved with stringent accuracy demands. From table 11 we conclude that the high-order PDIRKN methods are more efficient in the high accuracy range.

5.3.5. Semi-discrete partial differential equation

Consider the following initial-boundary-value problem (see [11]):

$$\frac{\partial^2 u}{\partial t^2} = \frac{4\pi^2 u^2}{1 + 2x - 2x^2}\frac{\partial^2 u}{\partial x^2} + 4\pi^2 u[4\cos^2(2\pi t) - 1], \quad 0 \leqslant t \leqslant 1, \quad 0 \leqslant x \leqslant 1, \tag{5.6}$$

with Dirichlet boundary conditions and exact solution $u = (1 + 2x - 2x^2)\cos(2\pi t)$. By using second-order symmetric spatial discretization on a uniform grid with mesh $\Delta x = 1/20$ we obtain a set of 19 ODEs. Table 12 shows that the

Table 11
Values of NCD and M for problem (5.5).

Methods	p	s^*	k	$M = 98$	$M = 196$	$M = 392$	$M = 783$
Nørsett$_3$	3	2	1	0.9	1.8	2.7	3.6
SFB$_3$	3	2	1	0.6	1.5	2.4	3.3
B$_3$	3	4	1	0.2	0.9	1.9	2.7
II - Radau IIA	3	3	2	0.9	2.0	2.9	4.0
Nørsett$_4$	4	3	1	0.7	1.5	2.7	4.0
SFB$_4$	4	3	1	1.2	2.4	3.6	4.8
II - Gauss–Legendre	4	3	2	1.7	3.2	4.5	5.9
CS$_5$	5	5	1	1.7	3.1	4.7	6.2
II - Radau IIA	5	4	3	2.1	3.8	5.6	7.3
CS$_6$	6	5	1	1.9	3.5	5.3	7.1
II - Gauss–Legendre	6	4	3	1.2	3.1	5.1	7.2
II - Radau IIA	7	5	4	1.1	3.3	5.9	8.5
II - Gauss–Legendre	8	5	4	1.1	3.2	5.6	8.0

Table 12
Values of NCD and M for problem (5.6).

Methods	p	s^*	k	$M = 200$	$M = 400$	$M = 800$	$M = 1600$
Nørsett_3	3	2	1	3.5	4.3	5.1	5.9
SFB_3	3	2	1	3.6	4.5	5.4	6.3
B_3	3	4	1	*	4.3	5.4	6.4
II - Radau IIA	3	3	2	3.7	5.1	6.0	6.8
Nørsett_4	4	3	1	3.4	4.2	5.2	5.9
SFB_4	4	3	1	5.5	6.4	7.6	8.8
II - Gauss–Legendre	4	3	2	5.0	6.3	7.8	9.2
CS_5	5	5	1	4.0	5.3	6.6	7.7
II - Radau IIA	5	4	3	4.2	5.2	6.3	7.7
CS_6	6	5	1	3.1	4.4	5.5	6.9
II - Gauss–Legendre	6	4	3	3.8	4.7	6.2	8.1
II - Radau IIA	7	5	4	*	4.7	6.0	8.5
II - Gauss–Legendre	8	5	4	3.6	4.4	5.5	7.0

PDIRKN methods are at least competitive and often more efficient than the sequential methods of the same order.

6. Concluding remarks

In this paper, we have shown that diagonally implicit iteration of fully implicit, pth-order RKN correctors leads to parallel DIRKN methods of order p with relatively few sequential stages. For Radau IIA and Gauss–Legendre correctors, the iteration parameters are determined in such a way that the methods are A-stable, strongly A-stable or L-stable. Numerical experiments clearly demonstrate the superiority of the parallel methods over most of the sequential SDIRKN methods available in the literature.

Acknowledgements

I like to thank Prof. Dr. P.J. van der Houwen and Dr. B.P. Sommeijer for their help during the preparation of this paper. I am also grateful to the referees for their useful comments.

References

[1] K. Burrage, A study of order reduction for semi-linear problems, Report, University of Auckland (1990).

[2] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations, Runge–Kutta and General Linear Methods* (Wiley, New York, 1987).

[3] J.R. Cash, Diagonally implicit Runge–Kutta formulae with error estimates, J. Inst. Math. Appl. 24 (1979) 293-301.

[4] 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.

[5] G.J. Cooper and A. Sayfy, Semiexplicit A-stable Runge–Kutta methods, Math. Comp. 33 (1979) 146, 541–556.

[6] M. Crouzeix, Sur l'approximation des équations différentielles opérationnelles linéaires par des méthodes de Runge–Kutta, Ph. D. Thesis, Université de Paris, France (1975).

[7] E. Fehlberg, Klassische Runge–Kutta–Nyström Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen $x'' = f(t, x)$, Computing 10 (1972) 305–315.

[8] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, Springer Series in Comp. Math., vol. 14 (Springer, Berlin, 1991).

[9] P.J. van der Houwen, B.P. Sommeijer and W. Couzy, Embedded diagonally implicit Runge–Kutta algorithms on parallel computers, Math. Comp. 58 (1992) 197, 135–159.

[10] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Stability of collocation-based Runge–Kutta–Nyström methods, BIT 31 (1991) 469–481.

[11] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Parallel diagonally implicit Runge–Kutta–Nyström methods, J. Appl. Numer. Math. 9 (1992) 111–131.

[12] A. Iserles and S.P. Nørsett, On the theory of parallel Runge–Kutta methods, IMA J. Numer. Anal. 10 (1990) 463–488.

[13] L. Kramarz, Stability of collocation methods for the numerical solution of $y'' = f(x, y)$, BIT 20 (1980) 215–222.

[14] Nguyen huu Cong, A-stable diagonally implicit Runge–Kutta–Nyström methods for parallel computers, Report NM-R9208, Centre for Mathematics and Computer Science, Amsterdam (1992).

[15] S.P. Nørsett, Semi-explicit Runge–Kutta methods, Report Mathematics and Computation No. 6/74, Dept. of Mathematics, University of Trondheim, Norway (1974).

[16] S.P. Nørsett and P.G. Thomsen, Embedded SDIRK-methods of basic order three, BIT 24 (1984) 634–646.

[17] P.W. Sharp, J.H. Fine and K. Burrage, Two-stage and three-stage diagonally implicit Runge–Kutta–Nyström methods of orders three and four, IMA J. Numer. Anal. 10 (1990) 489-504.

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

[19] K. Strehmel and R. Weiner, Nichtlineare Stabilität und Phasenuntersuchung adaptiver Nyström–Runge–Kutta Methoden, Computing 35 (1985) 325–344.

CHAPTER IV

Note on the performance of direct and indirect Runge-Kutta-Nyström methods

published in: *J. Comput. Appl. Math.* **45** (1993), 347-355

Letter Section

Note on the performance of direct and indirect Runge–Kutta–Nyström methods *

Nguyen huu Cong

Afdeling Numerieke Wiskunde, Centre for Mathematics and Computer Science, Amsterdam, Netherlands; and Faculty of Mathematics, Mechanics and Informatics, University of Hanoi, Viet Nam

Received 29 July 1992
Revised 12 October 1992

Abstract

Nguyen huu Cong, Note on the performance of direct and indirect Runge–Kutta–Nyström methods, Journal of Computational and Applied Mathematics 45 (1993) 347–355.

This paper deals with predictor-corrector iteration of Runge–Kutta–Nyström (RKN) methods for integrating initial-value problems for special second-order ordinary differential equations. We consider RKN correctors based on both direct and indirect collocation techniques. The paper focuses on the convergence factors and stability regions of the iterated RKN correctors. It turns out that the methods based on direct collocation RKN correctors possess smaller convergence factors than those based on indirect collocation RKN correctors. Both families of methods have sufficiently large stability boundaries for nonstiff problems.

Keywords: Runge–Kutta–Nyström methods; predictor-corrector methods.

1. Introduction

We will investigate a class of (explicit) predictor-corrector (PC) methods obtained by predictor-corrector iteration (or fixed-point iteration) of Runge–Kutta–Nyström correctors for

Correspondence to: Dr. Nguyen huu Cong, Afdeling Numerieke Wiskunde, Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands.

* These investigations were supported by the University of Amsterdam who provided the author with a research grant for spending a total of two years in Amsterdam.

0377-0427/93/$06.00 © 1993 – Elsevier Science Publishers B.V. All rights reserved

solving the initial-value problem (IVP) for nonstiff, special second-order ordinary differential equations (ODEs)

$$\frac{d^2y(t)}{dt^2} = f(y(t)). \tag{1.1}$$

The methods described in this note have the same nature as the PIRKN methods (parallel, iterated RKN methods) proposed in [5]. The present note is concerned with a comparison of the convergence factors and stability regions of PIRKN methods based on *direct* and *indirect* RKN methods. Indirect RKN methods are derived from RK methods for first-order ODEs (also used in [5]), whereas direct RKN methods are directly constructed for second-order ODEs (see [6]). The iterated methods will be referred to as *indirect* and *direct* PIRKN methods. It turned out that for direct PIRKN methods the convergence factors and error constants are smaller than those of indirect PIRKN methods, resulting in a better performance of the *direct* PIRKN methods. The stability of the two types of methods is comparable, in spite of the fact that the direct RKN correctors used are only conditionally stable, while the indirect RKN methods are unconditionally stable (see [6]). In two numerical experiments the superiority of direct PIRKN methods over indirect PIRKN methods is demonstrated.

For notational convenience, we assume that (1.1) is a scalar equation. However, all considerations below can be straightforwardly extended to a system of ODEs, and therefore, also to nonautonomous equations.

2. Direct PIRKN and indirect PIRKN methods

The starting point is a fully implicit s-stage RKN method. For a scalar equation, this method assumes the form

$$\boldsymbol{Y} = y_n\boldsymbol{e} + h\boldsymbol{c}y_n' + h^2A f(\boldsymbol{Y}), \quad y_{n+1} = y_n + hy_n' + h^2\boldsymbol{b}^{\mathrm{T}} f(\boldsymbol{Y}), \quad y_{n+1}' = y_n' + h\boldsymbol{d}^{\mathrm{T}} f(\boldsymbol{Y}), \tag{2.1}$$

where A is an $s \times s$ matrix, $\boldsymbol{b}$, $\boldsymbol{d}$, $\boldsymbol{c}$ are s-dimensional vectors, and $\boldsymbol{e}$ is the unit vector. Furthermore, we use the convention that for any given vector $\boldsymbol{v} = (v_j)$, $f(\boldsymbol{v})$ denotes the vector with entries $f(v_j)$.

Consider the following fixed-point iteration scheme (cf. [5]):

$$\boldsymbol{Y}^{(0)} = y_n\boldsymbol{e} + h\boldsymbol{c}y_n', \tag{2.2a}$$

$$\boldsymbol{Y}^{(j)} = y_n\boldsymbol{e} + h\boldsymbol{c}y_n' + h^2A f(\boldsymbol{Y}^{(j-1)}), \quad j = 1, \ldots, m, \tag{2.2b}$$

$$y_{n+1} = y_n + hy_n' + h^2\boldsymbol{b}^{\mathrm{T}} f(\boldsymbol{Y}^{(m)}), \qquad y_{n+1}' = y_n' + h\boldsymbol{d}^{\mathrm{T}} f(\boldsymbol{Y}^{(m)}). \tag{2.2c}$$

Notice that the s components of the vectors $\boldsymbol{Y}^{(j)}$ can be computed in parallel, provided that s processors are available, so that the computational time needed for one iteration of (2.2b) is equivalent to the time required to evaluate one right-hand side function on a sequential computer. Therefore, the method (2.2) was called a PIRKN method (parallel, iterated RKN method).

Regarding the prediction formula (2.2a) as the predictor method and (2.1) as the corrector method, (2.2) may be considered as a conventional PC method (in P(EC)^{m}E mode). Assuming that the function $f(y)$ is Lipschitz continuous and observing that (2.2a) defines a first-order predictor formula (i.e., $\boldsymbol{Y}^{(0)} - \boldsymbol{Y} = \mathrm{O}(h^2)$), the following theorem easily follows (see also [5,7]).

Theorem 2.1. *Let p be the order of the corrector method* (2.1). *Then on s-processor computers the PIRKN method* (2.2) *represents an explicit RKN method of order* $p^* = \min\{p, 2m+2\}$ *requiring* $m+1$ *sequential right-hand side evaluations per step.*

Remark. From Theorem 2.1, we see that by setting $m = [\frac{1}{2}(p-1)]$, $[\cdot]$ denoting the integer function, we have a PIRKN method of maximum order $p^* = p$ (order of the corrector) with only $[\frac{1}{2}(p+1)]$ sequential right-hand side evaluations per step.

In the following subsections, we concentrate on the convergence factors and stability regions of direct and indirect PIRKN methods. Specification of the parameters $(A, \boldsymbol{b}, \boldsymbol{d}, \boldsymbol{c})$ of the direct collocation corrector methods can be found in [4, Appendix].

2.1. Convergence

In actual integration, the number of iterations m is determined by some iteration strategy, rather than by order considerations. Therefore, it is of interest to know how the integration step affects the rate of convergence. The stepsize should be such that a reasonable convergence speed is achieved.

We shall determine the rate of convergence by using the test equation $y'' = \lambda y$, where λ runs through the eigenvalues of the Jacobian matrix $\partial f/\partial y$. For this equation, we obtain the iteration error equation

$$\boldsymbol{Y}^{(j)} - \boldsymbol{Y} = zA\big[\boldsymbol{Y}^{(j-1)} - \boldsymbol{Y}\big], \quad z := \lambda h^2, \ j = 1, \ldots, m.$$

Hence, with respect to the test equation, the rate of convergence is determined by the spectral radius $\rho(A)$ of the matrix A. We shall call $\rho(A)$ the *convergence factor* of the PIRKN method. Requiring that $\rho(zA) < 1$ leads us to the convergence condition

$$|z| < \frac{1}{\rho(A)} \quad \text{or} \quad h^2 \leqslant \frac{1}{\rho(A)\rho(\partial f/\partial y)}. \tag{2.3}$$

This convergence condition is of the same form as the stability condition associated with RKN methods. In analogy with the notion of the *stability boundary*, we shall call $1/\rho(A)$ the *convergence boundary*.

Let the RKN matrices generating the direct PIRKN methods and indirect PIRKN methods be denoted by A_{direct} and A_{indirect}, respectively. Table 2.1 lists the convergence boundaries $1/\rho(A_{\text{direct}})$ and $1/\rho(A_{\text{indirect}})$, and the reduction factors $\epsilon = \rho(A_{\text{direct}})/\rho(A_{\text{indirect}})$ of a number

Table 2.1
Convergence boundaries $1/\rho(A)$ and reduction factors ϵ

pth-order correctors	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	$p=9$	$p=10$
Indirect Gauss–Legendre		12.04		21.73		37.03		52.63
Direct Gauss–Legendre		20.83		34.48		55.54		76.92
Indirect Radau IIA	5.98		13.15		25.64		40.00	
Direct Radau IIA	10.41		20.40		37.03		55.55	
Reduction factors ϵ	0.57	0.58	0.64	0.63	0.69	0.66	0.72	0.68

of indirect PIRKN methods and their direct analogues. These figures show that the direct PIRKN methods have much larger convergence boundaries, and hence much smaller convergence factors, than indirect PIRKN methods of the same order.

2.2. Stability boundaries

The linear stability of the PIRKN method (2.2) is investigated by again using the model equation $y'' = \lambda y$, where λ runs through the eigenvalues of $\partial f/\partial y$. Applying (2.2) to the model equation, we obtain the recursion

$$V_{n+1} = M_m(z)V_n, \quad V_{n+1} = \begin{pmatrix} y_{n+1} \\ hy'_{n+1} \end{pmatrix},$$

with

$$M_m(z) := \begin{pmatrix} 1 + z\boldsymbol{b}^{\mathrm{T}}(I - zA)^{-1}\big(I - (zA)^{m+1}\big)\boldsymbol{e} & 1 + z\boldsymbol{b}^{\mathrm{T}}(I - zA)^{-1}\big(I - (zA)^{m+1}\big)\boldsymbol{c} \\ z\boldsymbol{d}^{\mathrm{T}}(I - zA)^{-1}\big(I - (zA)^{m+1}\big)\boldsymbol{e} & 1 + z\boldsymbol{d}^{\mathrm{T}}(I - zA)^{-1}\big(I - (zA)^{m+1}\big)\boldsymbol{c} \end{pmatrix}. \tag{2.4}$$

Similar to the stability considerations of RKN methods (cf. [6]), the matrix $M_m(z)$, which determines the stability of PIRKN methods, will be called the *amplification matrix*, its spectral radius $\rho(M_m(z))$ the *stability function*. For finite, given m, the stability intervals of PIRKN methods are given by

$$(-\beta(m), 0) := \{z: \rho(M_m(z)) < 1,\ z < 0\}.$$

From (2.4) we see that if z satisfies (2.3), then $M_m(z)$ converges to the amplification matrix $M(z)$ of the corrector as $m \to \infty$ (see [6]). Hence, the *asymptotic* stability interval for $m \to \infty$, $(-\beta(\infty), 0)$, contains the intersection on the negative z-axis of the stability interval $(-\beta_{\mathrm{corr}}, 0)$ of the generating corrector (see [6]) and the region of convergence in the complex z-plane defined by (2.3). For indirect PIRKN methods, where the corrector method is A-stable, the asymptotic stability region is not less than its region of convergence. For direct PIRKN methods, where the corrector method is conditionally stable with stability boundaries less than the convergence boundaries, the asymptotic stability region contains the stability region of the corrector method (see Table 2.1 for convergence boundaries and [4] for stability boundaries of direct collocation RKN correctors).

Table 2.2 lists the stability boundaries $\beta_{\mathrm{direct}}(m)$ and $\beta_{\mathrm{indirect}}(m)$ of direct PIRKN and indirect PIRKN methods, respectively. The stability boundaries corresponding to the minimal value of m needed to reach the order of the correctors are indicated in bold face. In actual computation, the stepsize h should of course be substantially smaller than allowed by condition (2.3), that is, we want $|z| < \alpha/\rho(A)$, where α is significantly smaller than 1. In Table 2.2, we added the value of α for which $0 \geqslant z \geqslant -\min\{\beta_{\mathrm{direct}}(\infty), \beta_{\mathrm{indirect}}(\infty)\}$. This value is denoted by α_{crit}. Table 2.2 shows that usually the stability boundaries of the indirect PIRKN methods are larger than those of the direct PIRKN methods. However, in actual computation, we also need fast convergence, so that the integration step may be much smaller than allowed by stability. The values of α_{crit} in the last column indicate that, as far as convergence is concerned, the direct methods are superior. By means of Table 2.2 we can select the number of iterations

Table 2.2
Stability boundaries β_{direct} and β_{indirect} for direct and indirect PIRKN methods

Generating corrector methods	p	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=\infty$	α_{crit}
Indirect Radau IIA	3	**4.94**	4.99	3.52	5.03	5.44	4.90	$\geqslant 5.98$	1.00
Direct Radau IIA	3	**6.00**	7.84	4.44	7.04	8.62	6.96	$\geqslant 8.61$	0.57
Indirect Gauss–Legendre	4	**12.00**	12.00	0.00	12.00	12.00	0.00	$\geqslant 12.04$	0.75
Direct Gauss–Legendre	4	**6.83**	0.00	0.00	8.57	0.00	0.00	$\geqslant 9.00$	0.43
Indirect Radau IIA	5	7.06	**2.19**	10.46	4.76	11.70	7.81	$\geqslant 13.15$	0.73
Direct Radau IIA	5	7.06	**0.49**	14.33	5.33	9.51	9.55	$\geqslant 9.55$	0.47
Indirect Gauss–Legendre	6	7.06	**0.00**	9.81	0.00	9.75	0.00	$\geqslant 21.73$	0.45
Direct Gauss–Legendre	6	7.06	**0.00**	18.77	0.00	9.80	0.00	$\geqslant 9.77$	0.28
Indirect Radau IIA	7	7.06	0.00	**9.50**	18.21	5.40	18.57	$\geqslant 25.64$	0.38
Direct Radau IIA	7	7.06	0.00	**9.51**	26.9	6.06	9.84	$\geqslant 9.84$	0.27
Indirect Gauss–Legendre	8	7.06	0.00	**9.51**	0.00	0.00	9.86	$\geqslant 37.03$	0.27
Direct Gauss–Legendre	8	7.06	0.00	**9.51**	0.00	0.37	9.86	$\geqslant 9.86$	0.18
Indirect Radau IIA	9	7.06	0.00	9.51	**0.21**	26.35	5.80	$\geqslant 40.00$	0.25
Direct Radau IIA	9	7.06	0.00	9.51	**0.03**	9.86	6.13	$\geqslant 9.86$	0.18
Indirect Gauss–Legendre	10	7.06	0.00	9.51	**0.00**	9.86	0.00	$\geqslant 52.63$	0.70
Direct Gauss–Legendre	10	7.06	0.00	9.51	**0.00**	9.86	0.01	$\geqslant 36.65$	0.48

needed to achieve an acceptable stability boundary (the corresponding boundaries are underlined). In this selection, the fifth-, sixth-, ninth- and tenth-order methods require one iteration more than the number of iterations needed to reach the order of the corrector (see Theorem 2.1).

2.3. The truncation error

Let us denote the step values associated with the corrector by u_{n+1} and u'_{n+1}, and define

$$E_m(z) := \begin{pmatrix} z\boldsymbol{b}^{\mathrm{T}}(zA)^{m+1}(I-zA)^{-1}\boldsymbol{e} & z\boldsymbol{b}^{\mathrm{T}}(zA)^{m+1}(I-zA)^{-1}\boldsymbol{c} \\ z\boldsymbol{d}^{\mathrm{T}}(zA)^{m+1}(I-zA)^{-1}\boldsymbol{e} & z\boldsymbol{d}^{\mathrm{T}}(zA)^{m+1}(I-zA)^{-1}\boldsymbol{c} \end{pmatrix},$$

$$\boldsymbol{w}_{n+1} = \begin{pmatrix} u_{n+1} \\ hu'_{n+1} \end{pmatrix}, \qquad \boldsymbol{v}_{n+1} = \begin{pmatrix} y_{n+1} \\ y'_{n+1} \end{pmatrix}.$$

It can be shown that $\boldsymbol{w}_{n+1} - \boldsymbol{v}_{n+1} = E_m \boldsymbol{v}_n$ (see [3,7]), so that the local truncation error of PIRKN methods can be written as the sum of the truncation error of the corrector and the iteration error of the PIRKN method:

$$\begin{pmatrix} y(t_{n+1}) \\ hy'(t_{n+1}) \end{pmatrix} - \boldsymbol{v}_{n+1} = \begin{pmatrix} y(t_{n+1}) \\ hy'(t_{n+1}) \end{pmatrix} - \boldsymbol{w}_{n+1} + E_m \boldsymbol{v}_n.$$

Our numerical experiments have shown that the truncation error of direct RKN correctors is smaller than that of indirect RKN correctors. Since the convergence factors of the direct PIRKN methods are also smaller than those of indirect PIRKN methods, there are two potential effects to expect that the truncation error of direct PIRKN methods is smaller than that of indirect PIRKN methods.

3. Numerical experiments

In this section we report numerical results obtained by direct and indirect PIRKN methods. The absolute error obtained at the end of the integration interval is presented in the form 10^{-d} (d may be interpreted as the number of correct decimal digits (NCD)). In order to see the efficiency of the direct PIRKN methods, we follow a dynamical strategy for determining the number of iterations in the successive steps. It seems natural to require that the iteration error is of the same order in h as the local error of the corrector. This leads us to the stopping criterion

$$\| \boldsymbol{Y}^{(m)} - \boldsymbol{Y}^{(m-1)} \|_\infty \leqslant C\, h^{p+1}, \tag{3.1}$$

where C is a problem- and method-dependent parameter. Furthermore, in the tables of results, N_{seq} denotes the total number of sequential right-hand side evaluations, and N_{steps} denotes the total number of integration steps. The following two problems possess exact solutions in closed form. Initial conditions are taken from the exact solutions.

3.1. Linear nonautonomous problem

As a first numerical test, we apply the various PIRKN methods to the linear problem (cf. [3, Problem 5.1])

$$\frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} = \begin{pmatrix} -2\alpha(t)+1 & -\alpha(t)+1 \\ 2(\alpha(t)-1) & \alpha(t)-2 \end{pmatrix} y(t), \quad \alpha(t) = \max(2\cos^2(t), \sin^2(t)),$$
$$0 \leqslant t \leqslant 20, \tag{3.2}$$

with exact solution $y(t) = (-\sin(t), 2\sin(t))^{\mathrm{T}}$. Table 3.1 clearly shows the improved accuracy of the direct PIRKN methods. In all experiments, the (averaged) number of iterations m needed to satisfy the stopping criterion (approximately) varies between $[\frac{1}{2}p]$ and $[\frac{1}{2}(p+1)]$.

Table 3.1 also shows that the number of iterations are for both the indirect and direct method the same, so that the smaller convergence factor of the direct methods does not seem to play a role. For problems which are locally of the form $y'' = \lambda y$ (such as problem (3.2)), this can be explained by considering the stopping criterion (3.1) more closely. Let us denote the step point values and the iterates corresponding to the direct and indirect PIRKN method by y_n, y'_n, $\boldsymbol{Y}^{(j)}$ and x_n, x'_n, $\boldsymbol{X}^{(j)}$, respectively, and define

$$\delta_m := \left[\boldsymbol{Y}^{(m)} - \boldsymbol{Y}^{(m-1)}\right] - \left[\boldsymbol{X}^{(m)} - \boldsymbol{X}^{(m-1)}\right],$$

where m is the actual number of iterations performed per step. If it turns out that the magnitude of δ_m is much smaller than that of the tolerance Ch^{p+1}, then this would explain that the direct and indirect PIRKN methods use the same number of iterations. Writing the recursion (2.2b) in the form

$$\boldsymbol{Y}^{(j)} = \left[I + zA + z^2A^2 + \cdots + z^jA^j\right](y_n\boldsymbol{e} + hy'_n\boldsymbol{c}), \tag{2.2b'}$$

and a similar expression for $\boldsymbol{X}^{(j)}$, we obtain

$$\delta_m = z^m(A_{\text{direct}})^m(y_n\boldsymbol{e} + hy'_n\boldsymbol{c}) - z^m(A_{\text{indirect}})^m(x_n\boldsymbol{e} + hx'_n\boldsymbol{c}).$$

Table 3.1
Values of NCD/N_{seq} for problem (3.2) obtained by PIRKN methods

Generating corrector methods	p	$N_{steps}=80$	$N_{steps}=160$	$N_{steps}=320$	$N_{steps}=640$	$N_{steps}=1280$	C
Indirect Radau IIA	3	2.1/160	3.0/320	3.9/640	4.8/1280	5.7/2560	10^4
Direct Radau IIA	3	2.5/160	3.5/320	4.4/640	5.3/1280	6.2/2560	10^4
Indirect Gauss–Legendre	4	4.0/227	5.3/476	6.5/958	7.7/1920	8.9/3840	10^1
Direct Gauss–Legendre	4	5.0/226	6.4/477	7.6/959	8.8/1920	10.0/3840	10^1
Indirect Radau IIA	5	5.3/238	6.8/480	8.3/1179	9.8/2511	11.3/5098	10^1
Direct Radau IIA	5	5.8/238	7.5/480	8.9/1179	10.4/2511	11.9/5098	10^1
Indirect Gauss–Legendre	6	7.4/318	9.2/640	11.0/1280	12.8/2560	14.6/5120	10^{-1}
Direct Gauss–Legendre	6	8.1/318	9.9/640	11.7/1280	13.5/2560	15.3/5120	10^{-1}
Indirect Radau IIA	7	8.7/320	10.9/737	13.0/1570	15.1/3184	17.2/6393	10^{-1}
Direct Radau IIA	7	9.1/320	11.6/737	13.7/1570	15.8/3184	17.9/6393	10^{-1}
Indirect Gauss–Legendre	8	11.0/395	13.4/799	15.8/1600	18.2/3200	20.6/6400	10^{-2}
Direct Gauss–Legendre	8	12.4/395	16.1/799	18.6/1600	21.3/3200	23.8/6400	10^{-2}
Indirect Radau IIA	9	13.5/400	15.2/926	17.9/1903	20.6/3830	23.4/7673	10^{-3}
Direct Radau IIA	9	12.7/400	16.0/926	18.7/1903	21.4/3830	24.2/7673	10^{-3}
Indirect Gauss–Legendre	10	14.9/477	17.8/959	20.8/1920	23.8/3840		10^{-3}
Direct Gauss–Legendre	10	16.6/477	18.6/959	21.6/1920	24.6/3840		10^{-3}

Hence, defining the defect

$$D_j(\boldsymbol{v}) := \left\| (A_{\text{direct}})^j \boldsymbol{v} - (A_{\text{indirect}})^j \boldsymbol{v} \right\|_\infty,$$

the quantity δ_m is bounded by

$$\begin{aligned} \| \delta_m \|_\infty &\leqslant |z^m| \left[|y_n| D_m(\boldsymbol{e}) + h |y_n'| D_m(\boldsymbol{c}) \right] \\ &\quad + |z^m| \left\| (A_{\text{indirect}})^m \left[(x_n - y_n)\boldsymbol{e} + h(x_n' - y_n')\boldsymbol{c} \right] \right\| \\ &\leqslant h^{2m} |\lambda^m| (|y_n| + h|y_n'|) D_m + \mathrm{O}(h^{p+2m}), \\ D_m &:= \max\{D_m(\boldsymbol{e}),\ D_m(\boldsymbol{c})\}, \end{aligned}$$

where λ runs through the spectrum of the Jacobian of the ODE. Ignoring the $\mathrm{O}(h^{p+2m})$ term, we conclude that the iteration processes in the direct and indirect methods are expected to satisfy the stopping criterion (3.1) after an equal number of iterations if $h^{2m} |\lambda^m| (|y_n| + h|y_n'|) D_m \ll C h^{p+1}$. For nonstiff problems (say $|\lambda| \leqslant 1$), this condition takes the form

$$D_m \ll \frac{C h^{p+1-2m}}{|y_n| + h|y_n'|}. \tag{3.3}$$

Table 3.2 lists the values of D_m for the correctors of Table 3.1 (notice that D_m vanishes for $m = 1$, and, if $p = 10$, also for $m = 2$; this is a direct consequence of the order condition $(q+1)(q+2)A\boldsymbol{c}^q = \boldsymbol{c}^{q+2}$ satisfied by RKN correctors derived from collocation, see [2, p.270]). By means of Table 3.2 it can be verified that the values of m, C and h used in Table 3.1 satisfy (3.3), explaining the identical performance of the iteration processes.

Table 3.2
Values of $D_m := \max\{D_m(e), D_m(c)\}$ for RKN correctors

Correctors	p	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$
Radau IIA	3	$2.5\cdot10^{-2}$	$1.7\cdot10^{-2}$	$8.2\cdot10^{-3}$	$1.5\cdot10^{-3}$	$1.2\cdot10^{-4}$	$6.0\cdot10^{-5}$	$4.5\cdot10^{-6}$	$1.2\cdot10^{-6}$
Gauss–Legendre	4	$8.0\cdot10^{-3}$	$4.0\cdot10^{-3}$	$6.9\cdot10^{-4}$	$1.5\cdot10^{-4}$	$9.1\cdot10^{-6}$	$3.7\cdot10^{-7}$	$1.0\cdot10^{-7}$	$6.2\cdot10^{-9}$
Radau IIA	5	0	$1.1\cdot10^{-3}$	$2.6\cdot10^{-4}$	$8.4\cdot10^{-5}$	$1.2\cdot10^{-5}$	$8.5\cdot10^{-7}$	$5.1\cdot10^{-8}$	$4.9\cdot10^{-9}$
Gauss–Legendre	6	0	$5.2\cdot10^{-4}$	$6.8\cdot10^{-5}$	$9.9\cdot10^{-6}$	$1.3\cdot10^{-6}$	$8.4\cdot10^{-8}$	$3.1\cdot10^{-9}$	$1.2\cdot10^{-10}$
Radau IIA	7	0	$4.8\cdot10^{-5}$	$2.8\cdot10^{-5}$	$2.2\cdot10^{-6}$	$3.7\cdot10^{-7}$	$5.2\cdot10^{-8}$	$3.1\cdot10^{-9}$	$1.1\cdot10^{-10}$
Gauss–Legendre	8	0	$2.5\cdot10^{-5}$	$1.2\cdot10^{-5}$	$5.9\cdot10^{-7}$	$3.2\cdot10^{-8}$	$5.6\cdot10^{-9}$	$3.5\cdot10^{-10}$	$1.2\cdot10^{-11}$
Radau IIA	9	0	0	$2.0\cdot10^{-6}$	$3.0\cdot10^{-7}$	$1.2\cdot10^{-8}$	$1.2\cdot10^{-9}$	$1.5\cdot10^{-10}$	$7.9\cdot10^{-12}$
Gauss–Legendre	10	0	0	$1.0\cdot10^{-6}$	$1.3\cdot10^{-7}$	$3.3\cdot10^{-9}$	$1.4\cdot10^{-10}$	$1.6\cdot10^{-11}$	$9.3\cdot10^{-13}$

Table 3.3
Values of NCD/N_{seq} for problem (3.4) obtained by PIRKN methods

Generating corrector methods	p	$N_{\text{steps}}=200$	$N_{\text{steps}}=400$	$N_{\text{steps}}=800$	$N_{\text{steps}}=1600$	$N_{\text{steps}}=3200$	C
Indirect Radau IIA	3	0.8/556	1.7/1182	2.6/2400	3.5/4800	4.4/9600	10^4
Direct Radau IIA	3	1.3/556	2.2/1182	3.1/2400	4.0/4800	4.9/9600	10^4
Indirect Gauss–Legendre	4	1.9/570	3.2/1208	4.4/2554	5.6/5353	6.8/11122	10^5
Direct Gauss–Legendre	4	2.7/570	3.9/1200	5.1/2510	6.3/5276	7.5/10991	10^5
Indirect Radau IIA	5	3.2/652	4.7/1411	6.2/2967	7.7/6147	9.2/12594	10^6
Direct Radau IIA	5	3.8/652	5.3/1411	6.8/2967	8.3/6147	9.8/12594	10^6
Indirect Gauss–Legendre	6	4.5/845	6.3/1765	8.1/3596	9.9/7301	11.7/14809	10^5
Direct Gauss–Legendre	6	5.3/841	7.2/1760	9.0/3585	10.8/7291	12.6/14790	10^5
Indirect Radau IIA	7	5.7/808	7.9/1760	10.0/3648	12.1/7482	14.2/15304	10^7
Direct Radau IIA	7	6.2/808	8.6/1760	10.7/3648	12.8/7482	14.9/15304	10^7
Indirect Gauss–Legendre	8	7.2/992	9.6/2060	12.0/4246	14.4/8684	16.8/17556	10^6
Direct Gauss–Legendre	8	8.1/991	10.5/2057	12.9/4244	15.3/8672	17.7/17549	10^6
Indirect Radau IIA	9	8.6/1036	11.3/2174	14.0/4479	16.8/9094	19.5/18422	10^7
Direct Radau IIA	9	9.4/1036	12.1/2174	14.8/4479	17.5/9094	20.2/18422	10^7
Indirect Gauss–Legendre	10	10.1/1207	13.1/2473	16.1/5054	19.1/10273	22.2/20826	10^6
Direct Gauss–Legendre	10	11.1/1207	14.1/2473	17.1/5052	20.1/10270	23.3/20825	10^6

3.2. Nonlinear Fehlberg problem

For the second numerical example, we consider the orbit equation (see [1])

$$\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}=\begin{pmatrix}-4t^2 & -2/r(t)\\ 2/r(t) & -4t^2\end{pmatrix}y(t),\quad r(t)=\sqrt{y_1^2(t)+y_2^2(t)}\,,\ \sqrt{\tfrac{1}{2}\pi}\leqslant t\leqslant 3\pi, \tag{3.4}$$

with exact solution $y(t)=(\cos(t^2),\sin(t^2))^{\mathrm{T}}$. The results are reported in Table 3.3. In this nonlinear problem, the superiority of direct PIRKN methods is once again demonstrated.

References

[1] E. Fehlberg, Klassische Runge–Kutta–Nyström Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen $x''=f(t,x)$, *Computing* **10** (1972) 305–315.

[2] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations, I. Nonstiff Problems* (Springer, Berlin, 1987).
[3] Nguyen huu Cong, A-stable diagonally implicit Runge–Kutta–Nyström methods for parallel computers, Report NM-R9208, Centre Math. Comput. Sci., Amsterdam, 1992; also: *Numer. Algorithms*, to appear.
[4] Nguyen huu Cong, Note on the performance of direct collocation-based parallel-iterated Runge–Kutta–Nyström methods, Report NM-R9214, Centre Math. Comput. Sci., Amsterdam, 1992.
[5] B.P. Sommeijer, Explicit, high-order Runge–Kutta–Nyström methods for parallel computers, *Appl. Numer. Math.*, to appear.
[6] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Stability of collocation-based Runge–Kutta–Nyström methods, *BIT* **31** (1991) 469–481.
[7] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Parallel diagonally implicit Runge–Kutta–Nyström methods, *Appl. Numer. Math.* **9** (2) (1992) 111–131.

CHAPTER V

Parallel block predictor-corrector methods of Runge-Kutta type

published in: *Appl. Numer. Math.* **13** (1993), 109-123

Parallel block predictor–corrector methods of Runge–Kutta type *

P.J. Van der Houwen and Nguyen huu Cong

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Received 24 September 1992
Revised 19 October 1992
Accepted 19 October 1992

Abstract

Van der Houwen, P.J. and Nguyen huu Cong, Parallel block predictor–corrector methods of Runge–Kutta type, Applied Numerical Mathematics 13 (1993) 109–123.

In this paper, we construct block predictor–corrector methods using Runge–Kutta correctors. Our approach consists of applying the predictor–corrector method not only at step points, but also at off-step points (block points), so that, in each step, a whole block of approximations to the exact solution is computed. In the next step, these approximations are used to obtain a high-order predictor formula by Lagrange or Hermite interpolation. By choosing the abscissas of the off-step points narrowly spaced, a much more accurately predicted value is obtained than by predictor formulas based on preceding step point values. Since the approximations at the off-step points to be computed in each step can be obtained in parallel, the sequential costs of these block predictor–corrector methods are comparable with those of a conventional predictor–corrector method. Furthermore, by using Runge–Kutta correctors, the predictor–corrector iteration scheme itself is also highly parallel. Application of these block predictor–corrector methods based on Lagrange–Gauss pairs to a few widely-used test problems reveals that the sequential costs are reduced by a factor ranging from 2 to 11 when compared with the best sequential methods.

Keywords. Numerical analysis; stability; parallelism.

1. Introduction

We will investigate a particular class of (explicit) predictor–corrector (PC) methods for solving the initial-value problem (IVP) for nonstiff, first-order differential equations

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

Correspondence to: P.J. van der Houwen, Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands.

* These investigations were supported by the University of Amsterdam who provided the second author with a research grant for spending a total of two years in Amsterdam.

0168-9274/93/$06.00 © 1993 – Elsevier Science Publishers B.V. All rights reserved

on parallel computers. It is our aim to improve the conventional PC methods by using parallel processors. At a first level, PC methods can be characterized by the values (p, k, β), where p is the order of the method, k is the number of right-hand side evaluations per step, and β characterizes the stability of the integration process, e.g., β may denote the real or imaginary stability boundaries β_{re} and β_{im} of the method. Evidently, we would like to have a PC method in which for given order p, the value of k is small and β is sufficiently large. The magnitude of β should take into account the costs per step, which leads us to the definition of the *effective or scaled* stability boundary β/k.

For *sequential* computers, the PC methods of Adams type belong to the most efficient nonstiff IVP solvers. The PECE mode of these methods are characterized by $[p, 2, \beta]$, where the effective stability boundaries $(\beta_{re}, \beta_{im})/2$ monotonically decrease from (1.20, 0.60) for $p = 3$ to (0.16, 0.09) for $p = 10$. Less popular are PC methods based on PC pairs consisting of "last step value predictors" and Runge–Kutta (RK) correctors. In $P(EC)^{p-1}E$ mode, these RK-type PC methods are characterized by $\{p, s(p-1)+1, \beta\}$, where s is the number of stages of the generating corrector. The effective stability boundaries $(\beta_{re}, \beta_{im})/(s(p-1)+1)$ strongly depend on the particular corrector chosen, but are extremely small for the higher-order RK correctors. The advantage of the RK-type PC methods is their one-step nature facilitating easy implementation and stepsize control. However, the relatively large number of right-hand side evaluations per step makes them unattractive from a computational point of view.

With the introduction of *parallel* computers, several authors have proposed *parallel* methods (mostly of PC type) and have tried to improve on the sequential PC methods. Parallel PC methods can again be characterized by $\{p, k, \beta\}$ if we define k as the *sequential* number of right-hand side evaluations per step, that is, the wall-clock time per step corresponds to the time needed to evaluate k right-hand side functions. With this meaning of k, the *effective* stability boundary on *parallel* computers can again be defined by β/k. Let us first consider the parallel implementation of the Adams PECE methods and RK-type PC methods in $P(EC)^{p-1}E$ mode. The Adams PECE methods are again characterized by $\{p, 2, \beta\}$ indicating that these methods do not have intrinsic parallelism. For future reference, the effective stability boundaries are listed in Table 1. If the RK-type PC methods are implemented on a parallel computer, then we can characterize them by $\{p, p, \beta\}$ which shows that the sequential costs are reduced by about a factor s. PC methods of this type have been discussed in [10,12,13,14,16]. An actual implementation, including a stepsize strategy, and a detailed performance analysis can be found in [10] where they were called *PIRK methods* (parallel iterated RK methods). The effective stability boundaries of PIRK methods using "last step value" predictors are listed in Table 1 (these methods possess stability boundaries that do not depend on the particular corrector chosen).

There have been several attempts to construct parallel methods without starting from a conventional sequential method [5,11,15,19]. For a number of these parallel methods, Table 2

Table 1
Effective stability boundaries $(\beta_{re}, \beta_{im})/k$ of PC methods

	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	$p=9$	$p=10$
Adams PECE	(1.20, 0.60)	(0.96, 0.58)	(0.70, 048)	(0.52, 0.35)	(0.39, 0.26)	(0.29, 0.18)	(0.22, 0.13)	(0.16, 0.09)
PIRK $(k=p)$	(0.84, 0.57)	(0.69, 0.70)	(0.63, 0.00)	(0.59, 0.00)	(0.56, 0.25)	(0.54, 0.42)	(0.52, 0.00)	(0.50, 0.00)

Table 2
Effective stability boundaries $(\beta_{re}, \beta_{im})/k$ of various parallel methods

Method	p	k	$\beta := (\beta_{re}, \beta_{im})$
Multiblock method [5, Methods {(2.7), (2.9)}]	3	2	(2.49, –)
BRK method [11, Method (4.1)]	3	1	(0.64, 0.65)
Miranker–Liniger method [15]	4	1	(0.50, 0.04)
Shampine–Watts–Worland [17,19]	4	2	(0.44, 0.58)
Multiblock method [5, Method {(2.11), (2.13)}]	4	2	(1.67, –)
Hermite–Gauss method [14]	4	2	–
BRK method [11, Method (4.7)]	4	1	(0.53, 0.05)
BRK method [11, Method {(4.3), (4.6)}]	4	2	(0.06, 0.05)
Cyclic multistep method [6, Table 2]	6	2	–
BRK method [11, Method {(4.12), (4.13)}]	6	2	(0.87, 0.29)
Cyclic multistep method [6, Table 2]	8	2	–
BRK method [11, Method {(4.14), (4.15)}]	8	2	(0.15, 0.07)

lists the corresponding $\{p, k, \beta/k\}$ values (if available). We remark that the cyclic multistep methods mentioned in this table refer to parallel modifications of the original methods of Donelson and Hansen [6].

A further increase of the amount of parallelism in step-by-step methods consists of computing parallel solution values not only at step points, but also at off-step points, so that, in each step, a whole block of approximations to the exact solution is computed. This approach was successfully used in [7] for obtaining reliable defect control in explicit RK methods. In this paper, we want to use this approach for constructing parallel PC methods where the value of k is substantially less than the order p and where, at the same time, the effective stability boundaries are acceptably large. In our case, the block of approximations is used to obtain a high-order predictor formula in the next step by some interpolation formula, e.g., Lagrange or Hermite interpolation. By choosing the abscissas of the off-step points narrowly spaced, we achieve much more accurately predicted values than can be obtained by predictor formulas based on preceding step point values. Moreover, the precise location of the off-step points can be used for minimizing the interpolation errors or for maximizing stability boundaries. Since the approximations at the off-step points to be computed in each step can be obtained in parallel, the sequential costs of this block PC method are equal to those of conventional PC methods. Furthermore, by using RK correctors, the PC iteration scheme itself is also highly parallel (cf. [10,13]). The RK-based block PC methods may be considered as block versions of the aforementioned PIRK methods, and will therefore be termed *block PIRK methods* (BPIRK methods).

We concentrated on BPIRK methods based on Lagrange predictors and Gauss corrections. The number of sequential function calls per step of Lagrange–Gauss BPIRK methods equals $k = m + 1$, where m denotes the number of iterations performed. Using p-point Lagrange interpolation predictors (i.e., the dimension of the block of approximations equals p resulting in predictor formulas of order $p - 1$) and pth-order Gauss correctors, we obtain a p-dimensional BPIRK method whose order equals p for all m (even $m = 0$). The abscissas of the off-step points were used for minimizing the predictor errors (in some sense, see Section 2.3). For these BPIRK methods, we computed the effective stability boundaries. It turned out that

Table 3
Effective stability boundaries $(\beta_{re}, \beta_{im})/k$ of BPIRK methods based on {Lagrange, Gauss} pairs

$p=4, k=3$	$p=6, k=2$	$p=8, k=1$	$p=10, k=4$
(0.42, 0.42)	(0.39, 0.15)	(0.39, 0.20)	(0.37, 0.36)

for $k \leqslant 4$, the scaled stability boundary β_{re}/k assumes values in the range [0.31, 0.44]. The values of β_{im}/k are less constant and are often quite small (see Table 6 in Section 2.4). Table 3 lists cases where k is minimal while both β_{re}/k and β_{im}/k are "substantial". These figures show that the requirement of "substantial" scaled stability boundaries makes the fourth- and tenth-order BPIRK methods relatively expensive. However, our numerical experiments reveal that in actual applications, the BPIRK methods of order four and ten already perform efficiently for $k = 1$ or $k = 2$. Hence, we conclude that minimizing the interpolation error leads to sufficiently stable methods requiring only one or two sequential function calls per step.

In Section 3, we present comparisons with sequential and parallel methods from the literature for two widely-used test examples, viz. FEHL: the Fehlberg problem (cf. [9, p. 174]) and JACB: the Jacobian elliptic functions problem (cf. [9, p. 236]). Let R be the factor by which the sequential costs (i.e., wall-clock time) are reduced by applying the BPIRK methods to obtain the same accuracy. Then, from a comparison with sequential methods, we find the reduction factors listed in Table 4.

These conclusions encourage us to pursue the analysis of BPIRK methods. In particular, we will concentrate on a performance analysis of other predictors and on stepsize strategies that exploit the special structure of BPIRK methods.

2. Block PIRK methods

For simplicity of notation, let the IVP be a scalar problem and let us consider the s-stage implicit RK method

$$\boldsymbol{U} = y_n \boldsymbol{e} + hA f(\boldsymbol{U}), \qquad y_{n+1} = y_n + h\boldsymbol{b}^{\mathrm{T}} f(\boldsymbol{U}), \tag{2.1}$$

where A is an $s \times s$ matrix, $\boldsymbol{b}$ is an s-dimensional vector, $\boldsymbol{e}$ is the unit vector, $\boldsymbol{U}$ is the stage vector with components U_i, and where $f(\boldsymbol{U})$ denotes the vector with components $f(U_j)$. Suppose that we apply (2.1) at t_n with distinct stepsizes $a_i h$, where $i = 1, \ldots, r$ and $a_1 = 1$.

Table 4
Reduction factors obtained by applying BPIRK methods

Problem	Method from the literature	BPIRK	R
FEHL	Dormand–Prince method of order 5	order 4	2
	Dormand–Prince method of order 8	order 8	5
JACB	Runge–Kutta–Hairer method of order 10	order 10	11

Then we obtain a block of r numerical approximations $y_{n+1,i}$ to the exact solution values $y(t_n + a_i h)$ defined by

$$\boldsymbol{U}_i = y_n \boldsymbol{e} + a_i h A f(\boldsymbol{U}_i), \quad y_{n+1,i} = y_n + a_i h \boldsymbol{b}^{\mathrm{T}} f(\boldsymbol{U}_i), \quad i = 1, \ldots, r. \tag{2.2}$$

Let

$$\boldsymbol{Y}_n := (y_{n,1}, \ldots, y_{n,r})^{\mathrm{T}}, \quad y_{n,1} := y_n, \tag{2.3}$$

and let us approximate the stage vectors $\boldsymbol{U}_i$ by

$$\boldsymbol{U}_i^{(0)} = V_i \boldsymbol{Y}_n + h W_i f(\boldsymbol{Y}_n), \quad i = 1, \ldots, r, \tag{2.4}$$

where V_i and W_i are $s \times r$ matrices determined by order conditions (see Section 2.1). Regarding (2.2) as correctors and (2.4) as predictors for the stage vectors, we arrive at the PC method (in PE(CE)^{m}E mode)

$$\begin{aligned} &\boldsymbol{U}_i^{(0)} = V_i \boldsymbol{Y}_n + h W_i f(\boldsymbol{Y}_n), \\ &\boldsymbol{U}_i^{(j)} = \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{Y}_n + a_i h A f\big(\boldsymbol{U}_i^{(j-1)}\big), \quad j = 1, \ldots, m, \\ &y_{n+1,i} = \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{Y}_n + a_i h \boldsymbol{b}^{\mathrm{T}} f\big(\boldsymbol{U}_i^{(m)}\big), \qquad \boldsymbol{Y}_{n+1} := (y_{n+1,1}, \ldots, y_{n+1,r})^{\mathrm{T}}, \end{aligned} \tag{2.5}$$

where $i = 1, \ldots, r$ and where $\boldsymbol{e}_1$ denotes the first unit vector. We may distinguish the following types of predictors:

Hermite:	$\boldsymbol{U}_i^{(0)} = V_i \boldsymbol{Y}_n + h W_i f(\boldsymbol{Y}_n)$,
Adams:	$\boldsymbol{U}_i^{(0)} = y_{n,1} \boldsymbol{e} + h W_i f(\boldsymbol{Y}_n)$,
Lagrange:	$\boldsymbol{U}_i^{(0)} = V_i \boldsymbol{Y}_n$,
Explicit BDF:	$\boldsymbol{U}_i^{(0)} = V_i \boldsymbol{Y}_n + h W_i f(y_{n,r} \boldsymbol{e})$.

In the case of a Lagrange predictor, the PE(CE)^{m}E mode reduces to P(CE)^{m}E mode. If $r = 1$, then (2.5) reduces to the PIRK method studied in [10]. We shall call (2.5) an *r-dimensional BPIRK method*.

Given the vector $\boldsymbol{Y}_n$, the r values $y_{n+1,i}$ can be computed in parallel and, on a second level, the components of the ith stage vector iterate $\boldsymbol{U}_i^{(j)}$ can also be evaluated in parallel. Hence, r-dimensional BPIRK methods based on s-stage RK correctors can be implemented on a computer possessing r parallel processors each of which is itself a parallel system with s parallel processors. The number of sequential evaluations of f per step of length h equals $k = m + 2$. If the matrices W_i vanish, then $k = m + 1$.

2.1. Order conditions for the predictor

The order conditions for the predictor to be of order q are derived by replacing both $\boldsymbol{Y}_n$ and $\boldsymbol{U}_i^{(0)}$ by exact solution values. On substitution of $y(t_{n-1}\boldsymbol{e} + h\boldsymbol{a})$ and $y(t_n \boldsymbol{e} + a_i h \boldsymbol{c})$, respectively, setting $\boldsymbol{c} := A\boldsymbol{e}$, and by requiring that the residue is of order $q + 1$ in h, we are led to the conditions

$$\begin{aligned} &y(t_n \boldsymbol{e} + a_i h \boldsymbol{c}) - V_i y(t_n \boldsymbol{e} + h(\boldsymbol{a} - \boldsymbol{e})) - h W_i y'(t_n \boldsymbol{e} + h(\boldsymbol{a} - \boldsymbol{e})) = \mathrm{O}(h^{q+1}), \\ &\quad i = 1, \ldots, r. \end{aligned} \tag{2.6}$$

Using the relation $y(te + hx) = \exp(hx\ \mathrm{d}/\mathrm{d}t)y(t)$, we can expand the left-hand side of (2.6) in powers of h:

$$\left[\exp\left(h(a_i\boldsymbol{c}+\boldsymbol{e})\frac{\mathrm{d}}{\mathrm{d}t}\right) - \left(V_i + W_i h\frac{\mathrm{d}}{\mathrm{d}t}\right)\exp\left(h\boldsymbol{a}\frac{\mathrm{d}}{\mathrm{d}t}\right)\right] y(t_{n-1})$$

$$= \sum_{j=0}^{q} \boldsymbol{C}_i^{(j)}\left(h\frac{\mathrm{d}}{\mathrm{d}t}\right)^j y(t_{n-1}) + \boldsymbol{C}_i^{(q+1)}\left(h\frac{\mathrm{d}}{\mathrm{d}t}\right)^{q+1} y(t^*) = \mathrm{O}(h^{q+1}), \tag{2.6'}$$

where t^* is a suitably chosen point in the interval containing the values $t_{n-1} + a_i h$, $i = 1, \ldots, r$, and where

$$\boldsymbol{C}_i^{(j)} := \frac{1}{j!}\left[(a_i\boldsymbol{c}+\boldsymbol{e})^j - V_i\boldsymbol{a}^j - jW_i\boldsymbol{a}^{j-1}\right] = \boldsymbol{0}, \qquad j = 0, 1, \ldots, q, \quad i = 1, \ldots, r. \tag{2.7a}$$

The $\boldsymbol{C}_i^{(j)}$, $i = 1, \ldots, r$, represent the error vectors of the predictor formula. From (2.6') we obtain the order conditions

$$\boldsymbol{C}_i^{(j)} = \boldsymbol{0}, \qquad j = 0, 1, \ldots, q, \quad i = 1, \ldots, r. \tag{2.7b}$$

The error vectors $\boldsymbol{C}_i^{(q+1)}$ are the *principal* error vectors of the predictor (it is assumed that $\boldsymbol{C}_i^{(q+1)}$ does not vanish).

If the conditions (2.7) are satisfied, then the iteration error associated with the stage vector and the step point value satisfy the order relations

$$\boldsymbol{U}_i - \boldsymbol{U}_i^{(m)} = \mathrm{O}(h^{q+m+1}),$$

$$v_{n+1,i} - y_{n+1,i} = a_i h\boldsymbol{b}^{\mathrm{T}}\left[f(\boldsymbol{U}_i) - f(\boldsymbol{U}_i^{(m)})\right] = \mathrm{O}(h^{q+m+2}),$$

where $v_{n+1,i}$ denote the exact corrector solutions. Thus, we have

Theorem 2.1. *If the conditions* (2.7) *are satisfied and if the generating corrector* (2.1) *is of order p, then the orders of the iteration error and the BPIRK method* (2.5) *are* $p_{\mathrm{iter}} = q + m + 1$ *and* $p^* := \min\{p, p_{\mathrm{iter}}\}$, *respectively.*

Let $q \geqslant r - 1$ and define the matrices

$$P_i := \left(\boldsymbol{e}, a_i\boldsymbol{c}+\boldsymbol{e}, (a_i\boldsymbol{c}+\boldsymbol{e})^2, \ldots, (a_i\boldsymbol{c}+\boldsymbol{e})^{r-1}\right), \quad P_i^* := \left((a_i\boldsymbol{c}+\boldsymbol{e})^r, \ldots, (a_i\boldsymbol{c}+\boldsymbol{e})^q\right),$$

$$Q := (\boldsymbol{e}, \boldsymbol{a}, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^{r-1}), \qquad Q^* := (\boldsymbol{a}^r, \ldots, \boldsymbol{a}^q),$$

$$R := (\boldsymbol{0}, \boldsymbol{e}, 2\boldsymbol{a}, 3\boldsymbol{a}^2, \ldots, (r-1)\boldsymbol{a}^{r-2}), \qquad R^* := (r\boldsymbol{a}^{r-1}, \ldots, q\boldsymbol{a}^{q-1}),$$

where the matrices P_i^*, Q^*, and R^* are assumed to be zero if $q = r - 1$. Then the conditions (2.7) can be presented in the form

$$P_i - V_i Q - W_i R = O, \quad P_i^* - V_i Q^* - W_i R^* = O, \qquad i = 1, \ldots, r. \tag{2.7'}$$

Since the abscissas a_j are assumed to be distinct, we may write

$$V_i = [P_i - W_i R]Q^{-1}, \quad P_i^* - [P_i - W_i R]Q^{-1}Q^* - W_i R^* = O, \qquad i = 1, \ldots, r.$$

Using Theorem 2.1, explicit expressions for the predictor matrices V_i and W_i can be derived. The following theorem presents these matrices for Lagrange predictors and Hermite predictors:

Theorem 2.2. *Let $\theta = 1$ and $\theta = 2$ respectively indicate the Lagrange and Hermite predictors. If*

$$q = \theta r - 1,$$

$$V_i = \left[P_i - (\theta - 1)W_i R\right] W Q^{-1},$$

$$W_i = (\theta - 1)\left[P_i Q^{-1} Q^* - P_i^*\right]\left[RQ^{-1}Q^* - R^*\right]^{-1}, \quad i = 1, \ldots, r,$$

then $p_{\text{iter}} = \theta r + m$, $p^* = \min\{p, p_{\text{iter}}\}$, and $k = m + \theta$, where $RQ^{-1}Q^* - R^*$ is assumed to be nonsingular.

In the application of BPIRK methods, we have two natural PC pairs, viz. Lagrange–Gauss pairs and Hermite–Radau pairs. The Lagrange–Gauss pairs have the advantage of (i) a maximal corrector-order for a given number of stages, (ii) no additional evaluations of f in the predictor (since we are aiming at a small number of iterations, say one or two, one extra f-evaluation substantially increases the total effort per step), and (iii) less round-off if the abscissas a_i are narrowly spaced. The disadvantage of Gauss correctors of being only A-stable is not relevant here, since BPIRK methods are designed for nonstiff problems, so that more stable correctors such as the L-stable Radau correctors are not needed. In the case of Radau correctors where the last component of the stage vector is identical to the step point value $y_{n+1,i}$, Hermite predictors are more natural because the additional f-evaluation needed in Hermite interpolation formulas is already available. An important advantage of using Hermite interpolation is the reduction of the number of processors needed for the implementation of BPIRK methods.

In this paper, we confine our considerations to Lagrange predictors and Gauss correctors. In the near future, we intend to compare BPIRK methods employing Lagrange, Hermite, Adams, and BDF predictors.

2.2. Region of convergence

In actual integration, the number of iterations m is determined by some iteration strategy, rather than by order considerations. Therefore, it is of interest to know how the integration step affects the rate of convergence. The stepsize should be such that a reasonable convergence speed is achieved.

We shall determine the convergence factor for the test equation $y' = \lambda y$, where λ runs through the eigenvalues of the Jacobian matrix $\partial f/\partial y$. For this equation, we obtain the iteration error equation

$$\boldsymbol{U}_i^{(j)} - \boldsymbol{U}_i = a_i z A\left[\boldsymbol{U}_i^{(j-1)} - \boldsymbol{U}_i\right], \quad z := h\lambda, \qquad j = 1, \ldots, m. \tag{2.8}$$

Table 5
Convergence boundaries $\gamma(\alpha)$

	$p=4$	$p=6$	$p=8$	$p=10$
Gauss–Legendre	3.46α	4.65α	6.06α	7.30α

Hence, with respect to the test equation, the convergence factor is defined by the spectral radius $\rho(a_i zA)$ of the iteration matrix $a_i zA$, $i=1,\ldots,r$. Requiring that $\rho(a_i zA)$ is less than a given number α leads us to the convergence condition

$$a_i h \leqslant \frac{\gamma(\alpha)}{\rho(\partial f/\partial y)}, \quad \gamma(\alpha) := \frac{\alpha}{\rho(A)}, \tag{2.9}$$

where $\gamma(\alpha)$ presents the convergence boundary of the method. In Table 5, the maximal convergence boundaries $\gamma(\alpha)$ are given for Gauss correctors of orders up to 10. In actual computation, the stepsize should of course be substantially smaller than allowed by $\gamma(1)$. Notice that for a given integration step h, the maximal damping factor is given by

$$\alpha = \frac{a_i h \rho(\partial f/\partial y)}{\gamma(1)},$$

so that the higher-order correctors listed in Table 5 give rise to faster convergence.

2.3. *On the choice of abscissas* a_i

The accuracy of Lagrange interpolation formulas improves if the abscissas of the interpolating values are more narrowly spaced. However, this will increase the magnitude of the entries of the matrix V_i, causing serious round-off errors. There are several ways to reduce this round-off effect: (i) multi-precision arithmetic, (ii) direct computation of the extrapolated values, and (iii) limitation of the spacing of the abscissas. The use of multi-precision arithmetic is the most simple remedy, but not always available and usually rather costly. Direct interpolation of the values $y_{n,1},\ldots,y_{n,r}$ requires in each step and for each component equation of the system of IVPs the solution of a linear system of dimension θr. Again, this option is rather costly. Probably, the most realistic option is a limitation on the minimal spacing of the abscissas a_i. In [7] where Hermite interpolation formulas were used for deriving reliable error estimates for defect control, it was found that on a Silicon Grafics Inc. Power Iris 4D/240S-64 machine with 15 digits precision, the abscissas should be separated by 0.2 in order to suppress rounding errors. For the more stable Lagrange interpolation formulas, we expect that slightly smaller spacings are still acceptable.

In order to derive further criteria for the choice of suitable values for the abscissas a_i, we need insight into the propagation of a perturbation ε of the block vector $\boldsymbol{Y}_n$ within a single step. We shall study this for the test equation $y' = \lambda y$. First we express $y_{n+1,i}$ in terms of $\boldsymbol{Y}_n$. Applying (2.5) and (2.8), we obtain the recursions

$$\boldsymbol{U}_i^{(0)} = [V_i + zW_i]\boldsymbol{Y}_n,$$
$$\boldsymbol{U}_i^{(j)} - \boldsymbol{U}_i = a_i zA\big[\boldsymbol{U}_i^{(j-1)} - \boldsymbol{U}_i\big), \quad j=1,\ldots,m.$$

Hence

$$\begin{aligned} y_{n+1,i} &= \boldsymbol{e}_1^{\mathrm{T}}\boldsymbol{Y}_n + a_i z\boldsymbol{b}^{\mathrm{T}}\big[\boldsymbol{U}_i^{(m)} - \boldsymbol{U}_i\big] + a_i z\boldsymbol{b}^{\mathrm{T}}\boldsymbol{U}_i \\ &= \boldsymbol{e}_1^{\mathrm{T}}\boldsymbol{Y}_n + a_i z\boldsymbol{b}^{\mathrm{T}}[a_i z A]^m\big[\boldsymbol{U}_i^{(0)} - \boldsymbol{U}_i\big] + a_i z\boldsymbol{b}^{\mathrm{T}}\boldsymbol{U}_i \\ &= \big(\boldsymbol{e}_1^{\mathrm{T}} + a_i z\boldsymbol{b}^{\mathrm{T}}[I - a_i z A]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\big)\boldsymbol{Y}_n \\ &\quad + a_i z\boldsymbol{b}^{\mathrm{T}}[a_i z A]^m\big[V_i + zW_i - [I - a_i z A]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\big]\boldsymbol{Y}_n \\ &= R(a_i z)\boldsymbol{e}_1^{\mathrm{T}}\boldsymbol{Y}_n + a_i z\boldsymbol{b}^{\mathrm{T}}[a_i z A]^m\big[V_i + zW_i - [I - a_i z A]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\big]\boldsymbol{Y}_n, \end{aligned} \tag{2.10}$$

where $i = 1, \ldots, r$, and $R(z)$ is the stability function of the RK corrector. Let us now replace $\boldsymbol{Y}_n$ by $\boldsymbol{Y}_n^* = \boldsymbol{Y}_n + \varepsilon$. Then, the perturbed value of $y_{n+1,i}$ is given by

$$\begin{aligned} y_{n+1,i}^* &= y_{n+1,i} + R(a_i z)\boldsymbol{e}_1^{\mathrm{T}}\varepsilon \\ &\quad + a_i z\boldsymbol{b}^{\mathrm{T}}[a_i z A]^m\big[V_i + zW_i - [I - a_i z A]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\big]\varepsilon. \end{aligned} \tag{2.10'}$$

This relation shows that the first component of the perturbation ε is amplified by a factor of O(1), whereas all other components are amplified by a factor of $O(h^{m+1})$.

Let us now return to the choice of the abscissas a_i. The values of the a_i influence the accuracy of the predicted stage values, and hence the accuracy of the block vectors $\boldsymbol{Y}_n$. Let ε represent the effect on $\boldsymbol{Y}_n$ of using inaccurate interpolation formulas in the preceding steps. Then, from the preceding discussion, we may conclude that the first component of ε is not damped. Since the components of the block vectors $\boldsymbol{Y}_n$ are calculated independently from the predicted stage values, it is important that the interpolation error corresponding to the predicted stage values used for the first component of the block vector are small. Thus, we should try to minimize the magnitude of the principal error vector $\boldsymbol{C}_1^{(q+1)}$.

In the case of Lagrange predictors where $q = r - 1$, we have to minimize the magnitude of $\boldsymbol{C}_1^{(r)}$. Although we may use (2.7a) for minimizing $\boldsymbol{C}_1^{(r)}$, it is more convenient to start with the usual expression for the remainder term in Lagrange interpolation formulas. For sufficiently differentiable functions $y(t)$, the r-point Lagrange interpolation formula can be written in the form (see e.g. [1, formulas 25.2.1–25.2.3])

$$\begin{aligned} y(t_n + \tau h) &= \sum_{i=1}^{r} L_i(\tau)\, y(t_{n-1} + a_i h) + C^{(r)}(\tau)\left(h\frac{\mathrm{d}}{\mathrm{d}t}\right)^r y(t^*), \\ C^{(r)}(\tau) &:= \frac{1}{r!}\prod_{i=1}^{r}[\tau + 1 - a_i], \end{aligned} \tag{2.11}$$

where $L_i(\tau)$ are the interpolation coefficients and t^* is a suitably chosen point in the interval containing the values $t_{n-1} + a_i h$, $i = 1, \ldots, r$. The principal error vectors of the Lagrange predictor formulas defined by Theorem 2.2 are given by $\boldsymbol{C}_i^{(r)} = C^{(r)}(\boldsymbol{c}a_i)$, $i = 1, \ldots, r$. Recalling that $a_1 = 1$, we are led to minimize the magnitude of the values

$$C^{(r)}(c_j) = \frac{1}{r!}\prod_{i=1}^{r}\big[c_j + 1 - a_i\big], \quad j = 1, \ldots, s.$$

Confining our considerations to block dimensions $r \geqslant s+1$, we set

$$a_1 = 1, \qquad a_i = 1 + c_{i-1}, \quad i = 2 \ldots, s+1. \tag{2.12a}$$

By this choice, the principal error vector $\boldsymbol{C}_1^{(r)}$ vanishes, so that now all inaccuracies introduced by the predictor formula are damped by a factor of $O(h^{m+1})$ (cf. (2.10′)). If $r > s+1$, then we have additional abscissas for improving the predictor formula. It is tempting to use these additional abscissas for reducing the magnitude of the other error vectors. From (2.11) it follows that the largest error constant (corresponding to the largest values of a_i and c_j) can be minimized by choosing the remaining abscissas close to 1. However, as already observed, the minimal spacing of the abscissas should be sufficiently large to avoid round-off. From (2.12a) it follows that the *averaged* spacing of the abscissas $a_1, \ldots, a_{s+1}$ is $1/(s+1)$ for correctors with $c_s \neq 1$ and $1/s$ otherwise, the *minimal* spacing being, in general, smaller. Therefore, it seems recommendable to choose the remaining abscissas outside the interval $[1,\ 1+c_s]$. In our numerical experiments, we have chosen the remaining abscissas such that averaged spacing equals that of the abscissas $a_1, \ldots, a_{s+1}$. This leads us to define the remaining abscissas according to

$$\begin{aligned} &\text{if } c_s \neq 1, \text{ then } a_i = \frac{s+i}{s+1}, \qquad i = s+2, \ldots, r, \\ &\qquad\qquad \text{else } a_i = \frac{s+i-1}{s}, \quad i = s+2, \ldots, r. \end{aligned} \tag{2.12b}$$

For Gauss correctors, the order p is equal to $2s$, resulting in an averaged spacing $2/(p+2)$. Recalling that the 15 digits experiments reported in [7] indicate that a minimal spacing of 0.2 is acceptable in the case of Hermite interpolation, we expect that on 15-digit computers and for orders up to $p = 10$, an averaged spacing of $2/(p+2)$ should be acceptable in the case of the more stable Lagrange interpolation formulas. We remark that the optimal location of the off-step points for defect control as derived in [7] is in the interval where the defect is to be computed, rather than advancing the current step point as in (2.12).

Finally, we remark that the abscissas defined by (2.12) enable us to develop various cheap strategies for stepsize control. For example, if $r \geqslant s+2$, then the difference $y_{n-1,s+2} - y_{n,1}$ can be used for obtaining an error estimate.

2.4. Stability

From (2.10) it follows that we may write

$$\boldsymbol{Y}_{n+1} = M_{mr}(z)\boldsymbol{Y}_n,$$

$$M_{mr}(z) := \begin{pmatrix} R(a_1 z)\boldsymbol{e}_1^{\mathrm{T}} + a_1 z \boldsymbol{b}^{\mathrm{T}}[a_1 zA]^m \left[V_1 + zW_1 - [I - a_1 zA]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\right] \\ \cdots \\ R(a_r z)\boldsymbol{e}_1^{\mathrm{T}} + a_r z \boldsymbol{b}^{\mathrm{T}}[a_r zA]^m \left[V_r + zW_r - [I - a_r zA]^{-1}\boldsymbol{e}\boldsymbol{e}_1^{\mathrm{T}}\right] \end{pmatrix}.$$

Evidently, the *asymptotic* stability region for $m \to \infty$ is the intersection in the z-plane of the stability region S_{corr} of the generating corrector and the region of convergence defined by the

Table 6
Effective stability boundaries $(\beta_{re}, \beta_{im})/k$ of BPIRK methods of order $p^* = p$ using Lagrange–Gauss pairs with $r = p$

p	$k=1$	$k=2$	$k=3$	$k=4$
4	(0.44, 0.00)	(0.40, 0.00)	(0.42, 0.42)	(0.37, 0.37)
6	(0.40, 0.08)	(0.39, 0.15)	(0.39, 0.03)	(0.38, 0.39)
8	(0.39, 0.20)	(0.38, 0.28)	(0.38, 0.35)	(0.37, 0.05)
10	(0.31, 0.00)	(0.37, 0.00)	(0.36, 0.03)	(0.37, 0.36)

points z where the eigenvalues of $a_i zA$ are within the unit disk. Hence, if the corrector is A-stable, then the asymptotic stability region in the left half-plane is completely determined by the region of convergence (see Table 5 for convergence boundaries).

For *finite* m, the stability regions are given by

$$S_{stab}(m, r) := \{z: \rho(M_{mr}(z)) < 1\}.$$

The associated real and imaginary stability boundaries β_{re} and β_{im} can be defined in the usual way.

Let us consider methods where $r = p$ and where the number of iterations is chosen dynamically by some iteration strategy. This type of methods use "maximal" block dimension r (in the sense that the order of the predictor equals that of the corrector) and iterate until a stable result is obtained assuming that the process converges. Again restricting our considerations to Lagrange–Gauss pairs, we obtain the results listed in Table 6. Because the effective, *real* stability boundaries are almost constant for all k, we may use $k = 1$ when only the real stability boundary plays a role. The *imaginary* stability boundaries show a less regular behaviour. BPIRK methods with $(r, p, k) = (4, 4, 3), (6, 6, 4), (8, 8, 2), (10, 10, 4)$ possess reasonably large effective real and imaginary stability boundaries (these cases are collected in Table 3). Notice that in all the cases the convergence regions contains the real and imaginary stability intervals, so that the integrations step will not be limited by convergence conditions, but rather by accuracy or stability conditions.

3. Numerical experiments

We tested accuracy and efficiency aspects of BPIRK methods based on Lagrange–Gauss pairs. All experiments are performed on a 28-digit computer, so that the effect of rounding errors is negligible. In Section 3.1, we will concentrate on the accuracy of the methods. In particular, the effective order and the influence of the number of iterations on the efficiency will be tested. In Section 3.2, we compare the BPIRK methods with block RK methods, and in Section 3.3 a number of tenth-order methods are compared. In all experiments, the abscissas a_i are defined according to (2.12).

The maximal absolute error obtained at $t = T$ is presented in the form $10^{-\Delta}$ (Δ may be interpreted as the number of correct decimal digits). Negative values of Δ are indicated by $*$. If the order of accuracy shown in the experiments equals the theoretical order p^*, then, on

halving the (fixed) stepsize, the number of correct decimal digits should increase by $0.3p^*$. Hence, the number of steps, denoted by N_{steps}, and Δ are related according to

$$N_{\text{steps}} = c2^{\Delta/(0.3p^*)},$$

where c is a constant depending on the problem. In order to verify this theoretical relation, we define the effective order

$$p_{\text{eff}} := \frac{\Delta(h) - \Delta(2h)}{0.3}. \tag{3.1}$$

In the first step, we always set $r = 1$ and $k = m + 1 = p$, where k is the number of *sequential* function calls per step. For the subsequent steps, we used either $r = 1$ (PIRK methods) or $r = p$, while k is specified in the tables of results. These methods will be denoted by PIRK(p, k) and BPIRK(p, k). The stepsize is chosen such that the total number of sequential function calls (approximately) equals a prescribed number N_{seq}. Since $N_{\text{seq}} = p + k(N_{\text{steps}} - 1)$, we have

$$N_{\text{steps}} = 1 + \left[\frac{N_{\text{seq}} - p}{p - r + 1} + \frac{1}{2}\right], \qquad h := \frac{T - t_0}{N_{\text{steps}}},$$

where $[\cdot]$ denotes the integer part function and T denotes the end point of the integration interval (the effect of the integer part operation causes that the actual number of sequential right-hand sides may be slightly different from the prescribed number N_{seq}).

3.1. Accuracy tests

Consider the often-used test problem of Fehlberg (cf. [9, p. 174])

$$\begin{aligned} y_1' &= 2ty_1 \log(\max\{y_2, 10^{-3}\}), & y_1(0) &= 1, \\ y_2' &= -2ty_2 \log(\max\{y_1, 10^{-3}\}), & y_2(0) &= \mathrm{e}, \end{aligned} \qquad 0 \leqslant t \leqslant T, \tag{3.2}$$

with exact solution

$$y_1(t) = \exp(\sin(t^2)), \quad y_2(t) = \exp(\cos(t^2)).$$

Tables 7 and 8 present results for the fourth- and eighth-order Gauss correctors. We listed values of Δ for prescribed numbers N_{seq} of sequential function calls and the effective orders

Table 7
Correct decimal digits at $t = T = 5$ for problem (3.2)

N_{seq}	DOPRI5	PIRK(4, 4)	BPIRK(4, k)		
			$k = 1$	$k = 2$	$k = 3$
240		1.2	3.5	3.5	2.4
480	2.9	2.7	5.1	4.8	3.7
960	4.6	3.9	6.7	6.0	4.9
1920	6.0	5.1	8.2	7.2	6.1
p_{eff}		4.0	5.0	4.0	4.0

Table 8
Correct decimal digits at $t = T = 5$ for problem (3.2)

N_{seq}	DOPRI8	PIRK(8, 8)	BPIRK(8, k)		
			$k = 1$	$k = 2$	$k = 3$
240		1.5	6.8	8.1	7.4
480		6.0	10.8	11.7	9.7
960	7.0	8.3	13.8	14.2	12.1
1920	9.9	10.3	16.9	16.7	14.5
p_{eff}		6.7	10.3	8.3	8.0

p_{eff} corresponding to the smallest stepsize h. In order to appreciate the accuracy of the BPIRK methods, we added the Δ-values produced by the PIRK methods and by the "best" sequential methods currently available. In Table 7 we included results obtained by the 5(4) Dormand–Prince RK pair (DOPRI5) taken from [9, Fig. 4.3], and in Table 8 we included results obtained by the 8(7) Dormand–Prince RK pair (DOPRI8) (see [10, Table 5]). Unlike the BPIRK results, the DOPRI results are obtained using a stepsize strategy, so that at first sight, a comparison may not be fair. However, the BPIRK methods can be provided with a stepsize strategy without additional costs per step (see [10]) and, for problem (3.1), stepsize strategies do not change the (N_{seq}, Δ) results very much. This may be concluded from a comparison of the PIRK(8, 8) results of Table 8 with the results reported in [10, Table 5] for the stepsize control version of PIRK(8, 8), i.e. the code PIRK8. Therefore, it seems fair to conclude that for the Fehlberg problem (3.1) the BPIRK(4, 1) method is at least a factor two faster than DOPRI5, and BPIRK(8, 2) beats DOPRI8 by at least a factor five.

3.2. Comparison with other parallel methods

In [11] parallel block Runge–Kutta methods (BRK methods) of orders up to 8 for nonstiff problems have been constructed and were shown to be highly efficient when compared with sequential methods. One of the test examples in [11] is the equation of motion of a rigid body without external forces (problem JACB in [9, p. 236]):

$$\begin{aligned} y_1' &= y_2 y_3, & y_1(0) &= 0, \\ y_2' &= -y_1 y_3, & y_2(0) &= 1, \quad 0 \leqslant t \leqslant T. \\ y_3' &= -0.51 y_1 y_2, & y_3(0) &= 1, \end{aligned} \tag{3.3}$$

Table 9 presents a comparison of the most efficient BRK methods with BPIRK methods of the same order. These (fixed-stepsize) results show that the BPIRK methods are about four times as efficient as the BRK methods. However, the BRK methods are all two-processor methods, whereas the BPIRK methods require $p^2/2$ processors.

3.3. Comparison of tenth-order methods

We repeat the (fixed-stepsize) experiment performed in [8], where a number of methods were compared by applying them to problem (3.3) with $T = 60$ and by counting the number of

Table 9
Comparison with methods from the literature for problem (3.3) with $T = 20$

Sequential right-hand sides N_{seq}	120	240	480	960	p_{eff}	$p^* = p$
BRK [10, PC pair (4.3)–(4.6) of Table 5.4]	*	3.3	4.7	6.0	4.3	4
BPIRK (4, 1)	4.3	5.8	7.2	8.7	5.0	4
BRK [10, PC pair (4.12)–(4.13) of Table 5.4]	3.2	5.1	6.9	8.7	6.0	6
BPIRK (6, 1)	6.8	9.3	11.3	13.4	7.0	6
BRK [10, PC pair (4.14)–(4.15)]	2.9	7.4	9.8	12.2	8.0	8
BPIRK (8, 2)	8.7	11.4	13.8	16.2	8.0	8

Table 10
Comparison with tenth-order methods from the literature for problem (3.3) at $T = 60$.

Method	k	p	N_{steps}	Δ	N_{seq}
Runge–Kutta–Curtis (cf. [8])	18	10	240	9.9	4320
Runge–Kutta–Hairer [8]	17	10	240	10.1	4080
PIRK(10, k) method [10, Table 4]	10	10	150	10.0	1560
BPIRK(10, k)	1	10	410	10.1	419
	2	10	190	10.1	389
	3	10	120	10.0	369

(sequential) function calls needed to obtain 10 digits accuracy. In Table 10, we reproduce the values given in [8,10] for a few tenth-order methods, and we added the results obtained by our tenth-order BPIRK method. From these results we conclude that the BPIRK(10, 3) method is about eleven times cheaper than the sequential Runge–Kutta–Hairer method and about four times cheaper than the PIRK(10, 10) method.

Acknowledgement

The authors are grateful to Dr. B.P. Sommeijer for his interest in our investigations and for his careful reading of the paper.

References

[1] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards Applied Mathematics Series 55 (Dover, New York, 1970).
[2] K. Burrage, The error behaviour of a general class of predictor–corrector methods, *Appl. Numer. Math.* 8 (1991) 201–216.
[3] K. Burrage, The search for the Holy Grail, or: Predictor–corrector methods for solving ODEIVPs *Appl. Numer. Math.* 11 (1993) 125–141.
[4] K. Burrage, Efficient block predictor-corrector methods with a small number of corrections, *J. Comput. Appl. Math.* 45 (1993) 139–150.

[5] M.T. Chu and H. Hamilton, Parallel solution of ODE's by multi-block methods, *SIAM J. Sci. Statist. Comput.* 8 (1987) 342–353.
[6] J. Donelson and E. Hansen, Cyclic composite multistep predictor-corrector methods, *SIAM Numer. Anal.* 8 (1971) 137–157.
[7] W.H. Enright and D.J. Highman, Parallel defect control, *BIT* 31 (1991) 647–663.
[8] E. Hairer, A Runge–Kutta method of order 10, *J. Inst. Math. Appl.* 21 (1978) 47–59.
[9] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems* (Springer, Berlin, 1987).
[10] P.J. van der Houwen and B.P. Sommeijer, Parallel iteration of high-order Runge–Kutta methods with stepsize control, *J. Comput. Appl. Math.* 29 (1990) 111–127.
[11] P.J. van der Houwen and B.P. Sommeijer, Block Runge–Kutta methods on parallel computers, *Z. Angew. Math. Mech.* 68 (1992) 3–10.
[12] K.R. Jackson and S.P. Nøsett, Parallel Runge–Kutta methods, Manuscript (1988).
[13] K.R. Jackson and S.P. Nørsett, The potential for parallelism in Runge–Kutta methods, Part I: RK formulas in standard form, Tech. Report No. 239/90, Department of Computer Science, University of Toronto, Toronto, Ont. (1990); Part II: RK predictor–corrector formulas (in preparation).
[14] I. Lie, Some aspects of parallel Runge–Kutta methods, Report No. 3/87, Division Numerical Mathematics, University of Trondheim, Norway (1987).
[15] W.L. Miranker and W. Liniger, Parallel methods for the numerical integration of ordinary differential equations, *Math. Comp.* 21 (1967) 303–320.
[16] S.P. Nørsett and H.H. Simonsen, Aspects of parallel Runge–Kutta methods, in: A. Bellen, C.W. Gear and E. Russo, eds., *Numerical Methods for Ordinary Differential Equations, Proceedings L'Aquilla 1987*, Lecture Notes in Mathematics 1386 (Springer, Berlin, 1989).
[17] L.F. Shampine and H.A. Watts, Block implicit one-step methods, *Math. Comp.* 23 (1969) 731–740.
[18] B.P. Sommeijer, Stability boundaries of block Runge–Kutta methods (in preparation).
[19] P.B. Worland, Parallel methods for the numerical solution of ordinary differential equations, *IEEE Trans. Comput.* 25 (1976) 1045–1048.

CHAPTER VI

Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial value problems

to appear in: *J. Comput. Appl. Math.* (1994)

Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial value problems

Nguyen huu Cong
CWI
P. O. Box 94079, 1090 GB Amsterdam, The Netherlands
&
Faculty of Mathematics, Mechanics and Informatics
University of Hanoi, Thuong Dinh, Dong Da, Hanoi, Vietnam

Abstract

This paper discusses parallel iteration schemes for collocation-based, symmetric Runge-Kutta (SRK) methods for solving nonstiff initial-value problems. Our main result is the derivation of four A-stable SRK corrector methods of orders 4, 6, 8, and 10 that optimize the rate of convergence when iterated by means of the highly parallel fixed point iteration process. The resulting PISRK method (parallel-iterated SRK method) shows considerably increased efficiency when compared with fixed point iteration process applied to Gauss-Legendre correctors.

1991 Mathematical Subject Classification: 65M12, 65M20
1991 CR Categories: G.1.7
Key Words and Phrases: Runge-Kutta methods, predictor-corrector methods, parallelism.

Note: These investigations were supported by the University of Amsterdam who provided the author with a research grant for spending a total of two years at CWI in Amsterdam.

1. Introduction

In the literature, a number of parallel numerical methods have been proposed to solve the initial-value problem (IVP) for the system of nonstiff first-order ordinary differential equations (ODEs)

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

Most of them are based on the highly parallel fixed point iteration (or predictor-corrector iteration) using a Runge-Kutta (RK) corrector already available in the literature (e.g. the Gauss-Legendre methods (cf. [6], [8], [11])). These correctors possess a high-order of accuracy and excellent stability properties for generating parallel methods. In the present paper, we propose a new class of symmetric RK methods of collocation type, to be called SRK methods, in which the abscissas are chosen such that the RK matrix has a minimized spectral radius. This property leads

to improved rate of convergence when applying the parallel iteration scheme. Like the conventional Gauss-Legendre methods, the resulting SRK methods are A-stable (cf. Subsection 3.3). However, the particular location of the abscissas decreases the order of accuracy of the SRK methods when compared with the Gauss-Legendre methods. To be more precise, in general, an s-stage SRK method is of order p = s or p = s+1 depending on whether s is even or odd, whereas an s-stage Gauss-Legendre method has order p = 2s. On a sequential computer, this would be a serious sacrifice, because for a given order p, the increased number of stages of the SRK correctors increases the computational work per iteration considerably. But on parallel computers, the *sequential* computational work is independent of the number of stages.

The parallel iterated SRK methods (PISRK methods) developed in this paper have the same predictor-corrector nature as the parallel iterated RK methods (PIRK methods) proposed in [6] and the block PIRK methods (BPIRK methods) of [5]. The predictor formula is based on extrapolation of preceding stage and steppoint values (cf. Subsection 3.1). Stability investigations reveal that the PISRK methods have sufficiently large stability regions for nonstiff problems (see Subsection 3.3). In Section 4, we compare the efficiency of PISRK methods with that of the PIRK and the BPIRK methods by means of a number of numerical experiments. These comparisons show that for a given order of accuracy, the efficiency of the PISRK methods is much higher than the efficiency of the PIRK methods and comparable with or superior to that of the BPIRK methods. If we take into account that PISRK methods need much less processors for their implementation than needed by the BPIRK methods, we conclude that the PISRK methods are more attractive than the BPIRK methods.

2. Symmetric RK methods

In this section, we construct various symmetric RK methods that will serve as correctors for the parallel iteration scheme. For simplicity of notation, we assume that the equation (1.1) is an autonomous, scalar equation. However, all considerations below can be straightforwardly extended to a system of ODEs, and therefore, also to nonautonomous equations. For autonomous, scalar equations, the general s-stage RK method then assumes the form

$$\text{(2.1)} \qquad \mathbf{Y}_n = \mathbf{e}y_n + hA f(\mathbf{Y}_n), \quad y_{n+1} = y_n + h\mathbf{b}^T f(\mathbf{Y}_n),$$

where A is an s-by-s matrix, $\mathbf{b}$ and $\mathbf{e}$ are s-dimensional vectors, $\mathbf{e}$ is the vector with unit entries, and $\mathbf{Y}_n$ is the stage vector corresponding to the n-th step. Furthermore, we use the convention that for any given vector $\mathbf{v} = (v_j)$, $f(\mathbf{v})$ denotes the vector with entries $f(v_j)$. From now on, we assume that the RK method (2.1) is a collocation method based on symmetrically distributed, distinct collocation points (that is, the

vector $\mathbf{c} = A\mathbf{e}$ is such that the abscissas $t_n + c_i h$ are symmetric with respect to $t_n + h/2$). These RK methods will be referred to as *SRK methods*. They form a special family of the class of symmetric RK methods (cf. [4] p. 217]).

The collocation principle ensures that the SRK method is of at least order $p = s$. The order can be increased by satisfying the orthogonality relation (cf., e.g. [4] p. 207)

$$\int_0^1 \prod_{i=1}^{s} (x - c_i)\, x^{j-1} dx = 0. \tag{2.2}$$

It is easily verified that this condition is automatically satisfied for $j = 1$ if s is odd. Thus, we have the result:

Theorem 2.1. *An s-stage SRK method is of order $p = s$ if s is even and of order $p = s + 1$ if s is odd.* []

This leads us to restrict our considerations to SRK methods with an odd number of stages.

In Section 3, it will turn out that it is convenient to iterate A-stable SRK correctors. Therefore, we now briefly discuss the A-stability of SRK methods. It is well known that RK methods are A-stable if the stability function is analytic in the left-half plane $C^- := \{z \in C: \mathrm{Re}(z) < 0\}$ (i.e., if the eigenvalues of the matrix A lie in the right-half plane $C^+ := \{z \in C: \mathrm{Re}(z) > 0\}$) and if it is bounded by 1 on the imaginary axis. Since SRK methods possess stability functions of modulus 1 along the imaginary axis, we have the result:

Theorem 2.2. *An s-stage SRK method is A-stable if A has its eigenvalues in the right-half plane.* []

3. Parallel-iterated SRK methods

Starting with the RK method (2.1), we consider the following fixed-point iteration scheme

$$\mathbf{Y}_n^{(j)} = \mathbf{e}y_n + hA\mathbf{f}(\mathbf{Y}_n^{(j-1)}),\ j = 1, \ldots, m, \tag{3.1b}$$

$$y_{n+1} = y_n + h\mathbf{b}^T\mathbf{f}(\mathbf{Y}_n^{(m)}). \tag{3.1c}$$

By using information from the preceding step, that is, the values of y_n and the stage vector $\mathbf{Y}_{n-1}^{(m)}$, we may define a predictor formula of the form

(3.1a) $\mathbf{Y}_n^{(0)} = V\mathbf{Y}_{n-1}^{(m)} + \mathbf{w}y_n$,

where V is an s-by-s matrix and $\mathbf{w}$ is an s-dimensional vector, both determined by order conditions (see Subsection 3.1). Notice that the s components of the vectors $f(\mathbf{Y}_n^{(j)})$ can be computed in parallel, provided that s processors are available. Hence, the computational time needed for one iteration of (3.1b) is equivalent to the time required to evaluate one right-hand side function f on a sequential computer. Thus, in (3.1) the number of sequential evaluations of f per step of length h equals m+1.

Regarding the prediction formula (3.1a) as the predictor method and (2.1) as the corrector method, (3.1) may be considered as a conventional predictor-corrector (PC) method (in $P(EC)^mE$ mode). This parallel PC method (3.1) is of the same nature as the PIRK methods (parallel iterated RK methods) considered in [6], and only differs by its predictor (3.1a) and the underlying SRK corrector. In analogy with the PIRK methods, the method (3.1) will be called a *PISRK method* (parallel iterated SRK method).

3.1. Order conditions for the predictor method

The order conditions for the predictor formula (3.1a) can be derived straightforwardly using Taylor expansions. We obtain an order s predictor if

(3.2) $$\frac{1}{j!}\left[(\mathbf{c} + \mathbf{e})^j - (V, \mathbf{w})\, \mathbf{a}^j \right] = \mathbf{0}, \quad \mathbf{a} := (\mathbf{c}^T, 1)^T, \; j = 0, 1, ..., s.$$

These conditions determine the matrix $(V, \mathbf{w})$. In order to express $(V, \mathbf{w})$ explicitly in terms of $\mathbf{c}$, we define the s-by-(s+1) and (s+1)-by-(s+1) matrices P and Q

(3.3a) $$P := (\mathbf{e}, (\mathbf{c} + \mathbf{e}), (\mathbf{c} + \mathbf{e})^2, ..., (\mathbf{c} + \mathbf{e})^s), \quad Q := (\mathbf{e}^*, \mathbf{a}, \mathbf{a}^2, ..., \mathbf{a}^s),$$

where $\mathbf{e}^*$ is the (s+1)-dimensional vector with unit entries. Condition (3.2) can be written in the form $P - (V,\mathbf{w})Q = O$, where O is s-by-(s+1) matrix with zero entries. Since the abscissas c_j are assumed to be distinct, we can write

(3.3b) $$(V, \mathbf{w}) = PQ^{-1}.$$

If (3.3) is satisfied, then the iteration errors associated with the stage vector and step point value satisfy the order relations

$$\mathbf{Y}_n - \mathbf{Y}_n^{(m)} = O(h^{m+s+1}),$$
$$u_{n+1} - y_{n+1} = h\, \mathbf{b}^T[f(\mathbf{Y}_n) - f(\mathbf{Y}_n^{(m)})] = O(h^{m+s+2}),$$

where u_{n+1} denotes the corrector solution at the step point t_{n+1}. The local truncation error of PISRK methods can be written as the sum of the truncation error of the SRK corrector and the iteration error of the PISRK method:

$$\begin{aligned} y(t_{n+1}) - y_{n+1} &= (y(t_{n+1}) - u_{n+1}) + (u_{n+1} - y_{n+1}) \\ &= O(h^{p+1}) + O(h^{m+s+2}) = O(h^{p^*+1}), \end{aligned}$$

where p is the order of the SRK corrector, $p^* = \min(p, m+s+1)$. Thus, we have:

Theorem 3.1. *If the generating SRK corrector method (2.1) is of order p and if $(V,\mathbf{w})$ is defined by (3.3), then on s-processor computers the PISRK method (3.1) represents an explicit method of order $p^* = \min\{p, m+s+1\}$ requiring $m+1$ sequential right-hand side evaluations per step.* []

3.2. Construction of SRK corrector methods

In this subsection we concentrate on SRK methods with an odd number of implicit stages (s = 3, 5, 7, 9) and we will construct SRK correctors such that the corresponding PISRK methods have maximized rates of convergence. The rate of convergence of PISRK methods is defined by using the model test equation $y' = \lambda y$, where λ runs through the eigenvalues of the Jacobian matrix $\partial\mathbf{f}/\partial\mathbf{y}$ (cf. [5], [9]). For this equation, we obtain the iteration error equation

$$\mathbf{Y}_n^{(j)} - \mathbf{Y}_n = zA\,[\mathbf{Y}_n^{(j-1)} - \mathbf{Y}_n], \quad z := \lambda h, \quad j = 1, \ldots, m.$$

Hence, with respect to the model test equation, the rate of convergence is determined by the spectral radius $\rho(A)$ of the matrix A. We shall call $\rho(A)$ the *convergence factor* of the PISRK method. By requiring that $\rho(zA) < 1$, we are led to the convergence condition

$$(3.4) \qquad |z| < \frac{1}{\rho(A)} \quad \text{or} \quad h < \frac{1}{\rho(A)\,\rho(\partial\mathbf{f}/\partial\mathbf{y})}.$$

We exploit the freedom in the choice of the collocation vector **c** for SRK correctors for minimizing the convergence factor $\rho(A)$, or equivalently, for maximizing the convergence region $\{z: \rho(zA) < 1\}$. By a numerical search, we found the collocation vectors and the corresponding convergence factors as listed in Table 3.1 (the specification of the parameters of the associated SRK corrector methods can be found in the Appendix to [10]). Table 3.1 also lists the convergence factors for the Gauss-Legendre based PIRK methods (we note that PIRK and BPIRK methods

have identical convergence factors). From these figures, we see that the convergence factors of the PISRK methods are substantially smaller than those of the PIRK methods of the same order.

Table 3.1. SRK collocation points and convergence factors of PISRK and (B)PIRK methods

Order	c_1	c_2	c_3	c_4	Convergence factors PISRK	(B)PIRK
p = 4	0.10300662				0.198	0.289
p = 6	0.04101173	0.21235714			0.123	0.215
p = 8	0.02180707	0.11383597	0.27544350		0.089	0.165
p = 10	0.01348800	0.07067122	0.17189713	0.31496835	0.070	0.137

3.3. Stability of PISRK methods

A numerical computation of the spectrum of the matrix A defining the SRK methods derived above shows that all the eigenvalues are lying in the right-half plane. Therefore, in view of Theorem 2.2, these SRK methods are A-stable. Evidently, when iterating until convergence, the stability region of the PISRK method is given by the intersection of its convergence region $\{z: |z| < 1 / \rho(A)\}$ and the stability region of the corrector. Hence, by virtue of the A-stability, we achieve that by maximizing the region of convergence, we have in fact maximized the region of stability. However, in actual computation, we often do not iterate until convergence, so that it is of interest to determine the stability regions as a function of m.

Applying (3.1) to the model test equation, we obtain

$$\begin{aligned} \text{(3.5a)} \qquad \mathbf{Y}_n^{(m)} &= \mathbf{e}y_n + zA\mathbf{Y}_n^{(m-1)} = (I + zA + (zA)^2 + \ldots + (zA)^{m-1})\mathbf{e}y_n \\ &\quad + (zA)^m\mathbf{Y}_n^{(0)} \\ &= (zA)^m V\mathbf{Y}_{n-1}^{(m)} + ((I - zA)^{-1}(I - (zA)^m)\mathbf{e} + (zA)^m\mathbf{w})y_n \end{aligned}$$

$$\begin{aligned} \text{(3.5b)} \qquad y_{n+1} &= y_n + z\mathbf{b}^T\mathbf{Y}_n^{(m)} \\ &= z\mathbf{b}^T(zA)^m V\mathbf{Y}_{n-1}^{(m)} \\ &\quad + (1 + z\mathbf{b}^T((zA)^m\mathbf{w} + (I - zA)^{-1}(I - (zA)^m)\mathbf{e}))y_n. \end{aligned}$$

From (3.5) we obtain the recursion

$$\begin{pmatrix} \mathbf{Y}_n^{(m)} \\ y_{n+1} \end{pmatrix} = M_m(z) \begin{pmatrix} \mathbf{Y}_{n-1}^{(m)} \\ y_n \end{pmatrix},$$

(3.6)

$$M_m(z) = \begin{pmatrix} (zA)^m V & (zA)^m \mathbf{w} + (I-zA)^{-1}(I-(zA)^m)\mathbf{e} \\ z\mathbf{b}^T(zA)^m V & 1 + z\mathbf{b}^T((zA)^m \mathbf{w} + (I-zA)^{-1}(I-(zA)^m)\mathbf{e}) \end{pmatrix}.$$

Similar to the stability considerations of block PIRK methods (cf. [5]), the (s+1)-by-(s+1) matrix $M_m(z)$ will be called the *amplification matrix*, and its spectral radius $\rho(M_m(z))$ the stability function. Notice that $\rho(M_m(z))$ converges to the absolute value of the stability function of the corrector method as $m \to \infty$, if z satisfies the convergence condition (3.4).

Using the familiar definition of the real and imaginary stability boundaries $\beta_{re}(m)$ and $\beta_{im}(m)$, we computed the stability pairs $(\beta_{re}(m), \beta_{im}(m))$ as listed in Table 3.2. We observe that for small m, the stability of PISRK methods is rather poor, but for $m \geq p/2$ (say), the stability boundaries are sufficiently large for nonstiff problems. Hence, already for relatively small numbers of iterations, the PISRK method is expected to perform stably.

Table 3.2. Stability pairs $(\beta_{re}(m), \beta_{im}(m))$ for various PISRK methods

Order	m = 1	m = 2	m = 3	m = 4	m = 5
p = 4	(0.04, 0.05)	**(0.43, 0.40)**	(0.96, 0.53)	(1.52, 0.42)	(2.13, 0.42)
p = 6	(0.00, 0.00)	(0.10, 0.10)	**(0.39, 0.40)**	(0.80, 0.82)	(1.25, 1.28)
p = 8	(0.00, 0.00)	(0.02, 0.02)	(0.15, 0.16)	**(0.42, 0.42)**	(0.77, 0.78)
p = 10	(0.00, 0.00)	(0.00, 0.00)	(0.06, 0.06)	(0.21, 0.21)	**(0.46, 0.46)**

4. Numerical experiments

In this section we report numerical results obtained by the PISRK, the PIRK and the BPIRK methods. The experiments were performed on a 28-digits arithmetic computer. The absolute error obtained at the end of integration interval is presented in the form 10^{-d} (d may be interpreted as the number of correct decimal digits (NCD)). We only compared methods of the *same order*, so that the accuracies are more or less comparable.

In order to see the efficiency of the PISRK, PIRK and BPIRK methods, we applied a dynamical strategy for determining the number of iterations in the successive steps. The stopping criterion is defined by

(4.1) $$\| \mathbf{Y}_n^{(m)} - \mathbf{Y}_n^{(m-1)} \|_\infty \leq \mathrm{TOL} = C\, h^p ,$$

where C is a problem- and method-dependent parameter, p is order of the corrector. Notice that by this criterion the iteration error has the same order in h as the underlying corrector. Furthermore, in the tables of results, N_{seq} denotes the total number of sequential right hand side evaluations, N_{steps} denotes the total number of integration steps, k denotes number of processors needed for implementation. In the first integration step, we used the trivial predictor formula $\mathbf{Y}_1^{(0)} = y_n \mathbf{e}$.

4.1. Fehlberg problem

As a first numerical test, we integrate the often-used Fehlberg problem (cf.[3])

(4.2)
$$\begin{aligned} y_1'(t) &= 2\, t\, y_1(t) \log(\max\{y_2(t), 10^{-3}\}), \quad y_1(0) = 1, \\ y_2'(t) &= -2\, t\, y_2(t) \log(\max\{y_1(t), 10^{-3}\}), \quad y_2(0) = e, \end{aligned} \qquad 0 \leq t \leq 5,$$

with exact solution $y_1(t) = \exp(\sin(t^2))$, $y_2(t) = \exp(\cos(t^2))$. The results listed in Table 4.1 show that PISRK is always superior to PIRK. In the low accuracy range, the convergence of the PISRK methods in the integration process is slower than that of the BPIRK methods. This may be explained by the fact that the stability region of the PISRK methods is not sufficiently large for low m-values (see Table 3.2). However, for a given stepsize, the accuracy of the PISRK results turns out to be higher than the accuracy of the BPIRK method, so that the efficiency of PISRK is at least as high as that of BPIRK. Particularly, in the high accuracy range, the superiority of the PISRK methods over the BPIRK methods is evident.

Table 4.1. Values of NCD / N_{seq} for problem (4.2) obtained by various parallel PC methods

Methods	k	p	N_{steps}=100	N_{steps}=200	N_{steps}=400	N_{steps}=800	N_{steps}=1600	C
BPIRK	8	4	3.2/200	4.3/406	5.4/844	6.5/1758	7.7/3759	10^3
PISRK	3	4	4.3/256	5.2/483	6.2/930	7.4/1820	8.7/3661	10^3
PIRK	2	4	2.7/392	4.0/842	5.2/1756	6.5/3650	7.7/7409	10^3
BPIRK	18	6	5.4/250	7.1/533	8.9/1150	10.7/2505	12.5/5317	10^3
PISRK	5	6	5.9/348	8.6/637	10.2/1194	12.2/2272	14.0/4398	10^3
PIRK	3	6	5.2/601	7.0/1245	8.9/2542	10.7/5199	12.5/10488	10^3
BPIRK	32	8	8.2/293	10.3/662	12.7/1432	15.0/2985	17.5/6233	10^3
PISRK	7	8	8.7/439	11.9/780	14.6/1439	17.3/2706	19.6/5116	10^3
PIRK	4	8	7.8/774	10.2/1603	12.6/3297	15.1/6674	17.5/13468	10^3
BPIRK	50	10	9.9/357	12.9/787	15.9/1710	18.9/3658	22.0/7540	10^3
PISRK	9	10	12.2/513	13.1/913	18.8/1654	21.7/3086	23.1/5919	10^3
PIRK	5	10	9.9/942	12.9/1947	15.9/3973	18.9/8134	22.0/16407	10^3

4.2. Orbit equation

Our second example is a well-known test problem in the RK-literature, viz. the orbit equation (cf. [7])

$$
\begin{aligned}
& y'_1(t) = y_3(t), && y_1(0) = 1 - \varepsilon, \\
& y'_2(t) = y_4(t), && y_2(0) = 0, \\
(4.3)\qquad & y'_3(t) = \frac{-\,y_1(t)}{(y_1^2(t) + y_2^2(t))^{3/2}}, && y_3(0) = 0, && 0 \le t \le 20, \\
& y'_4(t) = \frac{-\,y_2(t)}{(y_1^2(t) + y_2^2(t))^{3/2}}, && y_4(0) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}} && \varepsilon = \frac{3}{10}.
\end{aligned}
$$

The performance of the methods is shown by the results given in Table 4.2. Again PISRK is superior to PIRK, but now, the BPIRK methods are slightly more

efficient than PISRK in the low accuracy range. However, in the range of high accuracy, the PISRK methods are again superior to the BPIRK methods.

Table 4.2. Values of NCD / N_{seq} for problem (4.3) obtained by various parallel PC methods

Methods	k	p	N_{steps}=100	N_{steps}=200	N_{steps}=400	N_{steps}=800	N_{steps}=1600	C
BPIRK	8	4	3.0/203	4.6/404	5.0/880	6.1/1861	7.3/3924	10^0
PISRK	3	4	2.7/270	5.0/499	5.8/958	7.7/1880	8.9/3739	10^0
PIRK	2	4	3.1/441	3.7/905	4.9/1947	6.1/4000	7.3/8000	10^0
BPIRK	18	6	4.8/237	6.8/511	8.7/1106	10.4/2516	12.2/5185	10^{-1}
PISRK	5	6	5.3/373	7.9/659	10.0/1172	12.6/2221	14.0/4363	10^{-1}
PIRK	3	6	5.0/643	7.2/1302	8.9/2637	10.5/5499	12.3/11200	10^{-1}
BPIRK	32	8	7.2/276	9.7/632	12.2/1382	14.7/2956	17.2/6277	10^{-2}
PISRK	7	8	7.9/458	10.9/808	14.0/1436	16.6/2695	19.0/5063	10^{-2}
PIRK	4	8	7.6/837	10.4/1686	12.8/3397	15.0/6845	17.3/13827	10^{-2}
BPIRK	50	10	9.5/265	12.8/637	16.0/1469	19.0/3187	22.1/6957	10^{-2}
PISRK	9	10	9.8/538	14.1/930	17.0/1651	19.6/2990	23.9/5625	10^{-2}
PIRK	5	10	9.3/926	12.8/1926	16.3/3927	19.2/8226	22.2/16532	10^{-2}

Concluding remarks

This paper shows the performance of a special class of symmetric Runge-Kutta methods when they are used as corrector methods for generating parallel PC methods for nonstiff problems. By two examples, we have shown that for a given order p the resulting PISRK method is by far superior to the PIRK method (about a factor from 2 to 5). However, the number of necessary processors is a factor 2 - 2/p larger. This modest increase of processors seems to be a low price for the substantially increased efficiency. The PISRK method is roughly competitive with BPIRK in the low accuracy range, but clearly more efficient in the high accuracy range. But here, it is the PISRK method that needs less processors. In fact, the number of processors needed by BPIRK is a factor $p^2 / (2p - 2)$ larger.

Acknowledgement

The author is grateful to Prof. Dr. P.J. van der Houwen and Dr. B.P. Sommeijer for their help during the preparation of this note.

References

[1] **Butcher, J. C. (1987):** *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods,* John Wiley & Sons, Chichester - New York - Brisbane - Toronto - Singapore.

[2] **Dekker, K. and Verwer, J.G. (1984):** *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, Noth-Holland, Amsterdam - New York - Oxford.

[3] **Fehlberg, E. (1969):** *Classical fith-, sixth- and eighth-order Runge-Kutta formulas with stepsize control*, NASA Technical Report 287, 1968; extract published in Computing **4**, 93-106.

[4] **Hairer, E., Nørsett, S.P. & Wanner, G. (1987):** *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer-Verlag, Berlin.

[5] **Houwen, P.J. van der & Nguyen huu Cong (1993):** *Parallel block predictor-corrector methods of Runge-Kutta type,* Appl. Numer. Math. **13**, 109-123.

[6] **Houwen, P.J. van der & Sommeijer, B.P. (1990):** *Parallel iteration of high-order Runge-Kutta methods with stepsize control*, J. Comp. Appl. Math. **29**, 111-127.

[7] **Hull, T.E., Enright, W.H., Fellen, B.M. & Sedgwick, A.E. (1972):** *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal. **9**, 603-637.

[8] **Lie, I. (1987):** *Some aspects of parallel Runge-Kutta methods,* Report No. 3/87, Division Numerical Mathematics, University of Trondheim, Norway.

[9] **Nguyen huu Cong (1993):** *Note on the perfomance of direct and indirect Runge-Kutta-Nyström methods*, J. Comp. Appl. Math. **45**, 347-355.

[10] **Nguyen huu Cong (1993):** *Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial value problems*, Report NM-R9320, Centre for Mathematics and Computer Science, Amsterdam.

[11] **Nørsett, S.P. & Simonsen, H.H. (1989):** *Aspects of parallel Runge-Kutta methods,* in: A. Bellen, C.W. Gear and E. Russo (Eds.): Numerical Methods for Ordinary Differential Equations, Proceedings L'Aquila 1987, LNM 1386, Springer-Verlag, Berlin.

Summary

This thesis describes the construction and analysis of parallel numerical methods for the integration of initial-value problems for second-order differential equations (ODEs). In these methods, we consider so-called *parallelism across the method*, which means that the method itself possesses inherent parallelism so that its effectiveness is independent of the dimension of the ODE. On top of that, *parallelism across the problem* can be exploited to improve the efficiency for large systems of ODEs. Since the application of this second approach is rather straightforward, we confine ourselves to parallelism across the method.

The methods discussed in this thesis are based on the iterative solution of an implicit Runge-Kutta-Nyström (RKN) method, which will be called the corrector. With respect to the choice of the corrector we can distinguish two cases:

Firstly, the so-called *indirect* approach, by which we mean that the implicit RKN corrector is obtained by applying a classical Runge-Kutta (RK) method to the ODE written in first-order form. In this way, RKN correctors can be obtained with properties similar to those of RK methods (i.e., s-stage methods of order up to 2s and stage order s, combined with unconditional stability).

Secondly, and this is the topic of Chapter I, we can construct implicit RKN correctors based on the *direct* approach, by which we mean that the method is directly tuned to the special form of the second-order ODE. It turns out that direct, collocation-based RKN methods can be constructed which possess a stage order that is one higher than in the indirect approach. However, such direct methods loose the property of unconditional stability which is a useful property for the integration of *stiff* ODEs. Such problems are studied in the Chapters II and III.

In Chapter II we introduce the parallel, diagonally implicit iteration process to solve the underlying corrector. The resulting methods can be considered as diagonally implicit RKN (DIRKN) methods which require, *effectively*, only one LU-decomposition per step, and the solution of m (non)linear relations per step (m is the number of iterations). However, due to the special form of the iteration process, all linear algebra involved deals with systems of dimension equal to that of the ODE. In the Chapters II and III we study various startegies for choosing the free parameters in the iteration process. In Chapter II, these parameters are used to achieve fast convergence towards the corrector solution (i.e., minimization of the spectral radius of the iteration matrix), whereas the iteration process in Chapter III uses the (minimal) number of iterations to reach the order of the corrector; then, the free parameters are used to make the method unconditionally stable. Numerical results in both chapters show a substantially increased efficiency when compared with standard (sequential) DIRKN methods from the literature.

The remaining three chapters deal with *nonstiff* problems. Again, the starting point is a fully implicit corrector, but now we use fixed point iteration, which is highly parallel; as a consequence, the resulting method is *explicit*.

In Chapter IV the convergence of this iteration process for indirect and direct collocation-based RKN correctors is compared. It turns out that the "direct correctors" are to be preferred since they give rise to smaller concergence factors than the "indirect correctors" do. The stability regions of both families appear to be sufficiently large for the integration of nonstiff problems. Furthermore, to increase the efficiency, the method is provided with a dynamic iteration strategy (i.e., "stop the iteration as soon as the corrector is solved").

In the last two chapters of this thesis we discuss a few ideas, which have been worked out for *first-order* differential equations (we remark that the ideas described in these chapters can be extended to second-order differential equations as well as to stiff problems). Apart from calculating approximations in the step points only, the fixed point (RK-based) iteration process can equally well be used to obtain (in parallel) solution values at the off-step points. The advantage of this approach is that a whole block of approximations is obtained yielding accurate predictions to be used in the next step. As a result, the number of iterations decreases; however, since the amount of parallelism is increased, the required number of processors is much larger. By a number of experiments it is shown in Chapter V that the efficiency is increased by a factor ranging from 2 to 11 when compared with the best sequential methods.

Finally, in Chapter VI, we focus on the asymptotic convergence factor of the fixed point iteration process (to solve the RK corrector). By sacrificing the property of superconvergence, we construct symmetric, collocation-based RK correctors in which the free collocation points are used to achieve optimal convergence factors. To obtain the same order of accuracy, the number of processors is doubled, compared with fixed point iteration of traditional RK methods (such as the Gauss-Legendre methods). However, this is amply compensated by the increased efficiency.

Samenvatting

Dit proefschrift beschrijft de constructie en de analyse van parallelle, numerieke methoden voor de integratie van beginwaardeproblemen voor tweede-orde, gewone differentiaalvergelijkingen. In deze methoden beschouwen we het zogenaamde *parallellisme over de methode*, hetgeen betekent dat de methode zelf inherent parallellisme bezit zodat de effectiviteit onafhankelijk is van de dimensie van het stelsel differentiaalvergelijkingen. Bovendien kan *parallellisme over het probleem* benut worden om de efficiëntie voor grote stelsels te verhogen. Aangezien de toepassing van deze tweede aanpak tamelijk voor de hand liggend is, beperken we ons tot parallellisme over de methode.

De methoden die in dit proefschrift worden besproken, zijn gebaseerd op het iteratief oplossen van een impliciete Runge-Kutta-Nyström (RKN) methode, die de corrector zal worden genoemd. Met betrekking tot de keus van de corrector onderscheiden we twee gevallen:

Ten eerste is er de zogenaamde *indirecte* aanpak, waarmee we bedoelen dat de impliciete RKN corrector verkregen wordt door het toepassen van een klassieke Runge-Kutta (RK) methode op de differentiaalvergelijking, geschreven in eerste-orde vorm. Op deze manier kunnen RKN correctors verkregen worden die soortgelijke eigenschappen bezitten als RK methoden (zoals s-stage methoden waarvan de orde en de stage-orde respectievelijk de waarden 2s en s kunnen aannemen in combinatie met onvoorwaardelijke stabiliteit).

Voorts, en dit is het onderwerp van Hoofdstuk I, kunnen we impliciete RKN correctors construeren die gebaseerd zijn op de *directe* aanpak, waarmee we bedoelen dat de methode direct afgestemd is op de speciale vorm van de tweede-orde differentiaalvergelijking. Het blijkt dat directe, op het collocatieprincipe gebaseerde RKN methoden geconstrueerd kunnen worden die een stage-orde bezitten die één hoger is dan bij de indirecte aanpak mogelijk is. Echter, zulke directe methoden verliezen hun onvoorwaardelijke-stabiliteitseigenschap; deze eigenschap is nuttig voor de integratie van *stijve* differentiaalvergelijkingen. Zulke problemen worden bestudeerd in de Hoofdstukken II en III.

In Hoofdstuk II introduceren we de parallelle, diagonaal-impliciete iteratie-methode om de aan het proces ten grondslag liggende corrector op te lossen. De resulterende methode kan worden beschouwd als een diagonaal-impliciete RKN (DIRKN) methode die, *effectief*, slechts één LU-ontbinding per stap vergt, alsmede het oplossen van m (niet-)lineaire relaties per stap (m is het aantal iteraties). Echter, dankzij de speciale vorm van het iteratieproces, hebben we in het lineaire-algebra deel te maken met stelsels die een dimensie hebben gelijk aan die van de differentiaalvergelijking. In de Hoofdstukken II en III bestuderen we diverse

strategieën om de vrije parameters in het iteratieproces te kiezen. In Hoofdstuk II worden deze parameters gebruikt om een snelle convergentie naar de corrector-oplossing te bewerkstelligen (d.w.z., de spectraalstraal van de iteratiematrix wordt geminimaliseerd), terwijl het iteratieproces in Hoofdstuk III het (kleinste) aantal iteraties kiest dat nodig is om de orde van de corrector te bereiken; vervolgens worden de vrije parameters gebruikt om de methode onvoorwaardelijk stabiel te maken. Numerieke resultaten in beide hoofdstukken tonen aan dat de efficiëntie aanzienlijk is toegenomen, vergeleken met standaard (sequentiële) DIRKN methoden uit de literatuur.

De resterende drie hoofdstukken handelen over *niet-stijve* problemen. Opnieuw is een volledig impliciete corrector het uitgangspunt, maar nu gebruiken we fixed-point iteratie, hetgeen in hoge mate parallel is; als gevolg hiervan is de resulterende methode *expliciet*.

In Hoofdstuk IV wordt de convergentie van dit iteratieproces voor indirecte en directe collocatie-RKN correctors vergeleken. Het blijkt dat de "directe correctors" te verkiezen zijn aangezien deze aanleiding geven tot kleinere convergentiefactoren dan de "indirecte correctors". De stabiliteitsgebieden van beide families blijken voldoende groot te zijn voor de integratie van niet-stijve problemen. Bovendien is de methode, om de efficiëntie te vergroten, uitgerust met een dynamische iteratiestrategie (d.w.z., "stop met itereren zodra de corrector is opgelost").

In de laatste twee hoofdstukken van dit proefschrift bespreken we een paar ideeën die uitgewerkt zijn voor *eerste-orde* differentiaalvergelijkingen (opgemerkt zij, dat de ideeën zoals beschreven in deze hoofdstukken uitgebreid kunnen worden zowel voor tweede-orde differentiaalvergelijkingen als voor stijve problemen). Behalve uitsluitend benaderingen in de "step-points" te berekenen, kan het fixed-point (op RK-gebaseerde) iteratieproces tevens gebruikt worden om (parallel) oplossings-waarden in de tussenpunten te berekenen. Het voordeel van deze aanpak is dat een heel blok van benaderingen verkregen wordt, waarmee een nauwkeurige voorspelling voor de oplossing in de volgende stap gemaakt kan worden. Het gevolg hiervan is dat het aantal iteraties daalt; echter, het vereiste aantal processoren is veel groter, aangezien de "hoeveelheid" parallellisme is toegenomen. Een aantal experimenten in Hoofdstuk V toont aan dat de efficiëntie, vergeleken met de beste sequentiële methoden, is toegenomen met een factor variërend van 2 tot 11.

Tenslotte concentreren we ons, in Hoofdstuk VI, op de asymptotische convergentiefactor van het fixed-point iteratieproces (om de RK corrector op te lossen). Door de eigenschap van superconvergentie op te offeren, construeren we symmetrische, op collocatie gebaseerde RK correctors waarin de vrije collocatie-punten gebruikt worden om optimale convergentiefactoren te verkrijgen. Vergeleken met fixed-point iteratie van traditionele RK methoden (zoals de Gauss-Legendre methoden), moet het aantal processoren verdubbeld worden om dezelfde orde te bereiken. Dit wordt echter ruimschoots gecompenseerd door de vergrote efficiëntie.