Pergamon

Computers Math. Applic. Vol. 31, No. 2, pp. 111–121, 1996

0898-1221/96 $15.00 + 0.00

0898-1221(95)00198-0

Explicit Symmetric Runge-Kutta-Nyström Methods for Parallel Computers

N. H. CONG
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
and
Faculty of Mathematics, Mechanics and Informatics
University of Hanoi, Thuong Dinh, Dong Da, Hanoi, Vietnam

(Received April 1994; revised and accepted May 1995)

Abstract—In this paper, we are concerned with parallel predictor-corrector (PC) iteration of Runge-Kutta-Nyström (RKN) methods in $P(EC)^mE$ mode for integrating initial value problems for the special second-order equation $\mathbf{y}''(t) = \mathbf{f}(\mathbf{y}(t))$. We consider symmetric Runge-Kutta-Nyström (SRKN) corrector methods based on direct collocation techniques which optimize the rate of convergence of the PC iteration process. The resulting PISRKN methods (parallel iterated SRKN methods) are shown to be much more efficient when they are compared to the PC iteration process applied to the Gauss-Legendre RKN correctors.

Keywords—Runge-Kutta-Nyström methods, Predictor-corrector methods, Parallelism.

1. INTRODUCTION

There exists in the literature a number of explicit Runge-Kutta-Nyström (RKN) methods for the numerical solution of the nonstiff second-order initial-value problem (IVP)

$$\frac{d^2\mathbf{y}(t)}{dt^2} = \mathbf{f}(\mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}'(t_0) = \mathbf{y}'_0, \qquad t_0 \le t \le T, \tag{1.1}$$

where $\mathbf{y} : \mathbb{R} \to \mathbb{R}^N$, $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$. Methods up to order 10 can be found in [1–4]. In order to exploit the potential of vector and parallel computers, several classes of predictor-corrector (PC) methods based on correctors of RKN type have recently been considered in [5–7]. In the present paper, we propose a class of parallel PC methods on a new class of *Symmetric* RKN correctors (SRKN correctors). The new SRKN corrector method is constructed by optimizing the rate of convergence of the PC iteration process (see Section 3.2).

Having constructed optimal SRKN correctors, we apply the highly parallel PC iteration scheme. For a given order p, the resulting parallel iterated SRKN method (PISRKN method) has a larger number of processors, $p - 1$, when compared with $p/2$ for the PIRKN methods proposed in [5,7]. However, the rate of convergence of the PISRKN method is much better, so that its efficiency is increased. This increased efficiency is shown in Sections 4.1 and 4.2 where numerical results are presented by comparing the PISRKN methods with PIRKN methods and with sequential RKN methods available in the literature.

These investigations were partly supported by the University of Amsterdam, National Basis Research Program and Research Program B93-05-71.
The author is grateful to P.J. van der Houwen and B.P. Sommeijer for their help during the preparation of this paper. He is also grateful to the referees for their useful comments.

2. SYMMETRIC RKN METHODS

In this section, we construct the class of SRKN correctors that will be used in the parallel PC iteration scheme. We will start with a fully implicit s-stage direct collocation-based RKN method (see, e.g., [8]). By using the Kronecker product denoted by $\otimes$, for the system of equations (1.1), this method assumes the form (cf., e.g., [7])

$$\mathbf{Y}_n = \mathbf{e} \otimes \mathbf{u}_n + h\mathbf{c} \otimes \mathbf{u}'_n + h^2 (A \otimes I_N)\, \mathbf{F}(\mathbf{Y}_n), \tag{2.1a}$$

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{u}_n + h\mathbf{u}'_n + h^2 (\mathbf{b}^\top \otimes I_N)\, \mathbf{F}(\mathbf{Y}_n), \\ \mathbf{u}'_{n+1} &= \mathbf{u}'_n + h (\mathbf{d}^\top \otimes I_N)\, \mathbf{F}(\mathbf{Y}_n). \end{aligned} \tag{2.1b}$$

Here, A is an s-by-s matrix, I_N is the identity N-by-N matrix, $\mathbf{b}, \mathbf{c}, \mathbf{d}$ and $\mathbf{e}$ are s-dimensional vectors, $\mathbf{e}$ is the vector with unit entries, $\mathbf{Y}_n$ is the sN-dimensional stage vector with s vector components $\mathbf{Y}_{n,i}$ of dimension N corresponding to the n^{th} step, and $\mathbf{F}(\mathbf{Y}_n)$ is also the sN-dimensional vector $(\mathbf{f}(\mathbf{Y}_{n,i}))$, $i = 1, \ldots, s$. From now on, we assume that the components of the collocation vector $\mathbf{c} = (c_1, \ldots, c_s)^\top$ are symmetrically distributed; that is, c_j are symmetric with respect to $1/2$. To be more precise, let $c_1, \ldots, c_m$ be given and less than $1/2$; then all the c_i greater than $1/2$ are available as free parameters and defined by

$$\begin{aligned} &\text{for } s = 2m+1: \quad c_{m+1+i} = 1 - c_i \quad (i = 1, 2, \ldots, m), \quad c_{m+1} = 1/2, \\ &\text{for } s = 2m: \quad\quad c_{m+i} = 1 - c_i \quad (i = 1, 2, \ldots, m). \end{aligned}$$

These RKN methods will be referred to as *SRKN methods*.

It is known that a direct collocation-based s-stage RKN method is of at least step point order $p = s$ and stage order $r = s$ (see [8]). These orders can be increased (beyond s) if the orthogonality relation

$$\int_0^1 \prod_{i=1}^{s} (x - c_i) x^{j-1}\, dx = 0 \tag{2.2}$$

is satisfied for $j \geq 1$ (cf., e.g., [8, Theorems 3.3 and 3.4; 9, p. 207]). For the SRKN methods, condition (2.2) is satisfied for $j = 1$ if s is odd (see also [10]). This is readily seen by setting $x = \xi + 1/2$, $s = 2m + 1$; then condition (2.2) takes the form

$$\int_{-1/2}^{1/2} \phi(\xi)\, d\xi = 0, \tag{2.3}$$

where $\phi(\xi)$ is an odd function defined by $\phi(\xi) = \xi \prod_{i=1}^{m} (\xi + 1/2 - c_i)(\xi - 1/2 + c_i)$. Thus we have the following simple theorem.

THEOREM 2.1. *Let p and r be the step point order and stage order, respectively, of an s-stage SRKN method; then $p = r = s$ if s is even and $p = r = s + 1$ if s is odd.* ∎

This theorem leads us to restrict our considerations to the SRKN methods with an odd number of stages. SRKN methods are not A-stable (see [8]), but their stability intervals denoted by $(-\beta_{\text{corr}}, 0)$ are sufficiently large for nonstiff problems (see Section 3.3).

3. PARALLEL ITERATED SRKN METHODS

Using the SRKN method (2.1) as corrector and

$$\mathbf{Y}_n^{(0)} = (V \otimes I_N)\, \mathbf{Y}_{n-1}^{(m)} + \mathbf{w} \otimes \mathbf{y}_n \tag{3.1a}$$

as predictor, we arrive at the following PC iteration scheme (in $P(EC)^m E$ mode):

$$\mathbf{Y}_n^{(j)} = \mathbf{e} \otimes \mathbf{y}_n + h\mathbf{c} \otimes \mathbf{y}'_n + h^2 (A \otimes I_N)\, \mathbf{F}\left(\mathbf{Y}_n^{(j-1)}\right), \qquad j = 1, \ldots, m, \tag{3.1b}$$

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{y}'_n + h^2 (\mathbf{b}^\top \otimes I_N)\, \mathbf{F}\left(\mathbf{Y}_n^{(m)}\right), \\ \mathbf{y}'_{n+1} &= \mathbf{y}'_n + h (\mathbf{d}^\top \otimes I_N)\, \mathbf{F}\left(\mathbf{Y}_n^{(m)}\right), \end{aligned} \tag{3.1c}$$

where V is an s-by-s matrix and $\mathbf{w}$ is an s-dimensional vector, both determined by the order conditions (see Section 3.1). Notice that the s components of dimension N of the vectors $\mathbf{F}(\mathbf{Y}_n^{(j)})$ can be computed in parallel, provided that s processors are available. Hence, the computational costs of (3.1) are measured by the number of sequential right-hand side evaluations per step which equals $m+1$.

This parallel PC method (3.1) is of the same nature as the PIRKN methods (parallel iterated RKN methods) considered in [5,7], and only differs by its predictor (3.1a) and the underlying corrector. In analogy with the PIRKN methods, the method (3.1) will be called a *PISRKN method* (parallel iterated SRKN 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. Similar to that of the PISRK methods considered in [10], we obtain an s order predictor if

$$\frac{1}{j!}\left[(\mathbf{c}+\mathbf{e})^j - (V,\mathbf{w})\mathbf{a}^j\right] = \mathbf{0}, \qquad \mathbf{a} := \left(\mathbf{c}^\top, 1\right)^\top, \quad j = 1,2,\ldots,s. \tag{3.2}$$

The parameter matrix V and vector $\mathbf{w}$ in (3.1a) can be determined by (3.2). In order to express V and $\mathbf{w}$ explicitly in terms of $\mathbf{c}$, we define the s-by-s and $(s+1)$-by-$(s+1)$ matrices P and Q

$$P := (\mathbf{e}, (\mathbf{c}+\mathbf{e}), (\mathbf{c}+\mathbf{e})^2, \ldots, (\mathbf{c}+\mathbf{e})^s), \quad Q := (\mathbf{e}^*, \mathbf{a}, \mathbf{a}^2, \ldots, \mathbf{a}^s), \tag{3.3a}$$

where $\mathbf{e}^*$ is $(s+1)$-dimensional vector with unit entries. Using (3.3a), condition (3.2) can be written in the form

$$P - (V,\mathbf{w})Q = O,$$

where O is an s-by-$(s+1)$ matrix with zero entries. Since the abscissas c_j are assumed to be distinct, we can write

$$(V,\mathbf{w}) = PQ^{-1}. \tag{3.3b}$$

If (3.3) is satisfied, then the following order relations are obtained:

$$\begin{aligned}
\mathbf{Y}_n - \mathbf{Y}_n^{(m)} &= O\left(h^{2m+s+1}\right), \\
\mathbf{u}_{n+1} - \mathbf{y}_{n+1} &= h^2\left(\mathbf{b}^\top \otimes I_N\right)^\top \left[\mathbf{F}\left(\mathbf{Y}_n\right) - \mathbf{F}\left(\mathbf{Y}_n^{(m)}\right)\right] = O\left(h^{2m+s+3}\right), \\
\mathbf{u}'_{n+1} - \mathbf{y}'_{n+1} &= h\left(\mathbf{d}^\top \otimes I_N\right)\left[\mathbf{F}(\mathbf{Y}_n) - \mathbf{F}\left(\mathbf{Y}_n^{(m)}\right)\right] = O\left(h^{2m+s+2}\right).
\end{aligned}$$

Thus, we have the following theorem.

THEOREM 3.1. *If the corrector method (2.1) is of order p and if V and $\mathbf{w}$ are defined by (3.3), then the PISRKN method (3.1) represents an explicit method of order $p^* = \min(p, 2m+s+1)$ requiring $m+1$ sequential right-hand side evaluations per step.* ∎

3.2. Construction of SRKN Corrector Methods

We restrict our considerations to SRKN methods with an odd number of implicit stages ($s = 3,5,7,9$) and we will construct SRKN correctors such that the corresponding PISRKN methods have an optimized rate of convergence. This rate of convergence of PISRKN methods is defined by using the model test equation $y'' = \lambda y$, where λ runs through the eigenvalues of the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ (cf., e.g., [5]). For this equation, we obtain the iteration error equation

$$\mathbf{Y}_n^{(j)} - \mathbf{Y}_n = zA\left[\mathbf{Y}_n^{(j-1)} - \mathbf{Y}_n\right], \qquad z := \lambda h^2, \quad j = 1,\ldots,m.$$

Hence, with respect to the model test equation, the rate of convergence is determined by the spectral radius $\rho(A)$ of the matrix A. By requiring that $\rho(zA) < 1$, we are led to the convergence condition

$$|z| < \frac{1}{\rho(A)} \quad \text{or} \quad h^2 < \frac{1}{\rho(A)\rho\left(\frac{\partial \mathbf{f}}{\partial y}\right)}. \tag{3.4}$$

We shall call $\rho(A)$ the *convergence factor* and $1/\rho(A)$ the *convergence boundary* of the PISRKN method. We exploit the freedom in the choice of the collocation vector $\mathbf{c}$ for SRKN correctors for minimizing the convergence factor $\rho(A)$, or equivalently, for minimizing the convergence region $\{z : z < 1/\rho(A)\}$. By a numerical search, we found the collocation vectors and the corresponding convergence factors as listed in Table 1 (the specification of the parameters of the associated SRKN corrector methods can be found in the Appendix). Table 1 also lists the convergence factors for the Gauss-Legendre based indirect and direct PIRKN methods proposed in [7,10]. From this table, we see that the convergence factors of the PISRKN methods are substantially smaller than those of the PIRKN methods of the same order.

Table 1. SRKN collocation points and convergence factors of PIRKN and PISRKN methods.

Order	c_1	c_2	c_3	c_4	Convergence factors		
					Indirect PIRKN	Direct PIRKN	PISRKN
$p=4$	0.10575846				0.083	0.048	0.025
$p=6$	0.04282436	0.21758171			0.046	0.029	0.011
$p=8$	0.02294808	0.11836119	0.28107352		0.027	0.018	0.006
$p=10$	0.01532451	0.07956500	0.19035553	0.33824665	0.019	0.013	0.004

We remark that the free parameters c_i could be chosen so that the error coefficients were as small as possible. This may lead to other collocation vector $\mathbf{c}$. As a consequence, the collocation points found in Table 1 aim at fast convergence of the PISRKN methods.

3.3. Stability of PISRKN Methods

The stability of the PISRKN method (3.1) is investigated by again using the model test equation $y''(t) = \lambda y(t)$, where λ is assumed to be negative. Applying (3.1) to the model test equation, we obtain

$$\begin{aligned}
\mathbf{Y}_n^{(m)} &= \mathbf{e}y_n + \mathbf{c}hy'_n + zA\mathbf{Y}_n^{(m-1)} = \left(I + zA + \cdots + z^{m-1}A^{m-1}\right)\left(\mathbf{e}y_n + \mathbf{c}hy'_n\right) + z^m A^m \mathbf{Y}_n^{(0)} \\
&= z^m A^m V \mathbf{Y}_{n-1}^{(m)} + \left((I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{e} + z^m A^m \mathbf{w}\right) y_n \qquad (3.5a) \\
&\quad + (I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{c}hy'_n, \\
y_{n+1} &= y_n + hy'_n + z\mathbf{b}^\top \mathbf{Y}_n^{(m)} \\
&= z^{m+1}\mathbf{b}^\top A^m V \mathbf{Y}_{n-1}^{(m)} + \left(1 + z\mathbf{b}^\top\left(z^m A^m \mathbf{w} + (I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{e}\right)\right) y_n \qquad (3.5b) \\
&\quad + \left(1 + z\mathbf{b}^\top (I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{c}\right) hy'_n, \\
hy'_{n+1} &= hy'_n + z\mathbf{d}^\top \mathbf{Y}_n^{(m)} \\
&= z^{m+1}\mathbf{d}^\top A^m V \mathbf{Y}_{n-1}^{(m)} + z\mathbf{d}^\top\left(z^m A^m \mathbf{w} + (I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{e}\right) y_n \qquad (3.5c) \\
&\quad + \left(1 + z\mathbf{d}^\top (I - zA)^{-1}\left(I - z^m A^m\right)\mathbf{c}\right) hy'_n.
\end{aligned}$$

By defining $\mathbf{X}_n^{(m)} := ((\mathbf{Y}_{n-1}^{(m)})^\top, y_n, hy'_n)^\top$, (3.5) leads us to the recursion

$$\mathbf{X}_{n+1}^{(m)} = M_m(z)\mathbf{X}_n^{(m)}, \tag{3.6}$$

$$M_m(z) = \begin{pmatrix}
z^m A^m V & (I - zA)^{-1}(I - z^m A^m)\mathbf{e} + z^m A^m \mathbf{w} & (I - zA)^{-1}(I - z^m A^m)\mathbf{c} \\
z^{m+1}\mathbf{b}^\top A^m V & 1 + z\mathbf{b}^\top (z^m A^m \mathbf{w} + (I - zA)^{-1}(I - z^m A^m)\mathbf{e}) & 1 + z\mathbf{b}^\top (I - zA)^{-1}(I - z^m A^m)\mathbf{c} \\
z^{m+1}\mathbf{d}^\top A^m V & z\mathbf{d}^\top (z^m A^m \mathbf{w} + (I - zA)^{-1}(I - z^m A^m)\mathbf{e}) & 1 + z\mathbf{d}^\top (I - zA)^{-1}(I - z^m A^m)\mathbf{c}
\end{pmatrix}.$$

Similar to the stability considerations of block PIRK and PISRK methods (cf. [10,11]), the $(s+2)$-by-$(s+2)$ matrix $M_m(z)$ which determines the stability of the PISRKN methods will be called the *amplification matrix*, and its spectral radius $\rho(M_m(z))$ the *stability function*. For finite, given m, the stability intervals of PISRKN methods are defined by

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

From (3.6) we see that if z satisfies the convergence condition (3.4), then the stability function of the PISRKN method $\rho(M_m(z))$ converges to the stability function of the SRKN corrector method as $m \to \infty$ (cf. [8]). Hence, the *asymptotic* stability interval of the PISRKN method for $m \to \infty, (-\beta(\infty), 0)$, contains the intersection on the negative z-axis of the stability interval $(-\beta_{\text{corr}}, 0)$ of the SRKN corrector (see [8]) and its region of convergence defined by (3.4). Since the convergence boundary of the PISRKN method $1/\rho(A)$ is much larger than the stability boundary β_{corr} of the corresponding SRKN corrector, the asymptotic stability interval of the PISRKN method coincides with the stability interval of the SRKN corrector. Table 2 lists the stability boundaries $\beta(m)$ of the resulting PISRKN methods. We observe that these stability boundaries are sufficiently large for nonstiff problems.

Table 2. Stability boundaries for various PISRKN methods.

Order	$\beta(1)$	$\beta(2)$	$\beta(3)$	$\beta(4)$	$\beta(5)$	$\cdots$	$\beta(\infty) = \beta_{\text{corr}}$
$p = 4$	0.21	2.66	4.15	4.96	5.49	...	9.76
$p = 6$	0.01	0.69	3.18	7.16	10.03	...	9.86
$p = 8$	0.00	0.19	1.47	4.44	9.18	...	39.47
$p = 10$	0.00	0.05	0.68	2.64	6.24	...	79.65

4. NUMERICAL EXPERIMENTS

In this section, we present numerical experiments applied to various parallel PC methods as well as sequential methods. In order to see the efficiency of the PC methods, we applied a dynamical strategy for determining the number of iterations in the successive steps

$$\left\| \mathbf{Y}_n^{(m)} - \mathbf{Y}_n^{(m-1)} \right\|_\infty \leq TOL = Ch^{p-1}, \tag{4.1}$$

where p is the order of the corrector method, and C is a parameter depending on the method and on the problem (cf. [5,6,10]). Notice that by the criterion (4.2) the iteration error of the PC method is of the same order in h as the underlying corrector. For PISRKN methods, in the first integration step, we used the trivial predictor formula $\mathbf{Y}_1^{(0)} = \mathbf{e} \otimes \mathbf{y}_n + h\mathbf{c} \otimes \mathbf{y}'_n$.

4.1. Comparison with Parallel Methods

In this section, we report numerical results obtained by the best parallel methods available in the literature, the PIRKN proposed in [7] and the PISRKN methods considered in this paper. 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)). 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 three problems possess exact solutions in closed form. Initial conditions are taken from the exact solutions.

4.1.1. Linear nonautonomous problem

As a first numerical test, we integrate the linear problem (cf. [5,12])

$$\frac{d^2\mathbf{y}(t)}{dt^2} = \begin{pmatrix} -2\alpha(t)+1 & -\alpha(t)+1 \\ 2(\alpha(t)-1) & \alpha(t)-2 \end{pmatrix} \mathbf{y}(t), \quad \alpha(t) = \max\left(2\cos^2(t), \sin^2(t)\right), \tag{4.2}$$
$$0 \leq t \leq 20,$$

Table 3. Values of NCD/N_{seq} for problem (4.3) obtained by various p^{th}-order parallel PC methods.

p^{th}-order PC methods	p	$N_{\text{steps}} = 80$	$N_{\text{steps}} = 160$	$N_{\text{steps}} = 320$	$N_{\text{steps}} = 640$	$N_{\text{steps}} = 1280$	C
PIRKN	4	4.0/237	5.3/477	6.5/958	7.7/1919	8.9/3836	10^{-1}
PISRKN	4	5.5/161	7.1/321	8.1/641	9.3/1281	10.5/2561	10^{-1}
PIRKN	6	7.4/320	9.2/640	11.0/1280	12.8/2559	14.6/5119	10^{-3}
PISRKN	6	9.3/232	11.0/433	12.9/704	15.0/1282	16.9/2562	10^{-3}
PIRKN	8	11.0/399	13.4/799	15.8/1600	18.2/3198	20.6/6398	10^{-4}
PISRKN	8	11.9/222	14.5/400	17.3/783	19.7/1410	23.2/2563	10^{-4}
PIRKN	10	13.3/436	18.0/921	20.9/1881	23.8/3803		10^{-4}
PISRKN	10	14.0/245	17.0/439	21.0/801	24.1/1497		10^{-4}

with exact solution $\mathbf{y}(t) = (-\sin(t), 2\sin(t))^{\mathsf{T}}$, by various p^{th}-order PC methods. The results listed in Table 3 clearly show that the PISRKN methods are by far superior to the PIRKN methods of the same order.

4.1.2. Orbit problem

For the second numerical example, we consider the often-used orbit problem (or nonlinear Fehlberg problem) (cf., e.g., [1,2,13,14])

$$\frac{d^2\mathbf{y}(t)}{dt^2} = \begin{pmatrix} -4t^2 & -\frac{2}{r(t)} \\ \frac{2}{r(t)} & -4t^2 \end{pmatrix} \mathbf{y}(t), \qquad r(t) = \sqrt{y_1^2(t) + y_2^2(t)}, \quad \sqrt{\frac{\pi}{2}} \le t \le 10. \tag{4.3}$$

The exact solution is given by $\mathbf{y}(t) = (\cos(t^2), \sin(t^2))^{\mathsf{T}}$. The results are reported in Table 4. Again the PISRKN methods are superior to the PIRKN methods of the same order.

Table 4. Values of NCD/N_{seq} for problem (4.4) obtained by various p^{th}-order parallel PC methods.

p^{th}-order PC methods	p	$N_{\text{steps}} = 200$	$N_{\text{steps}} = 400$	$N_{\text{steps}} = 800$	$N_{\text{steps}} = 1600$	$N_{\text{steps}} = 3200$	C
PIRKN	4	1.6/591	2.8/1197	4.0/2400	5.2/4800	6.4/9600	10^2
PISRKN	4	3.2/481	4.7/918	5.9/1693	7.0/3201	8.2/6401	10^2
PIRKN	6	4.0/775	5.8/1532	7.6/3096	9.4/6257	11.2/12648	10^3
PISRKN	6	6.8/526	8.0/1001	9.7/1887	11.5/3514	13.4/6553	10^3
PIRKN	8	6.6/1022	9.0/2032	11.5/4028	13.9/7966	16.3/15725	10^3
PISRKN	8	9.1/628	11.7/1094	14.5/2107	17.0/4076	19.4/7781	10^3
PIRKN	10	9.4/1234	12.4/2458	15.5/4893	18.5/9734	21.5/19332	10^3
PISRKN	10	12.4/699	15.4/1244	18.7/2226	22.3/4295		10^3

4.1.3. Newton's equations of motion problem

The third example is the two-body gravitational problem for Newton's equation of motion (see [15, p. 245]):

$$\begin{aligned} &\frac{d^2y_1(t)}{dt^2} = -\frac{y_1(t)}{(r(t))^3}, \quad \frac{d^2y_2(t)}{dt^2} = -\frac{y_2(t)}{(r(t))^3}, \qquad 0 \le t \le 20, \\ &y_1(0) = 1-\varepsilon, \quad y_2(0) = 0, \quad y_1'(0) = 0, \quad y_2'(0) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}, \end{aligned} \tag{4.4}$$

where $r(t) = \sqrt{y_1^2(t) + y_2^2(t)}$.

The solution components are $y_1(t) = \cos(u) - \varepsilon$, $y_2(t) = \sqrt{(1+\varepsilon)(1-\varepsilon)}\sin(u)$, where u is the solution of Kepler's equation $t = u - \varepsilon\sin(u)$ and ε denotes the eccentricity of the orbit. In this example we set $\varepsilon = 0.3$. The results are listed in Table 5. In this example, we observe the same increased efficiency of the PISRKN methods as in two previous examples.

Table 5. Values of NCD/N_{seq} for problem (4.5) obtained by various p^{th}-order parallel PC methods.

p^{th}-order PC methods	p	$N_{\text{steps}} = 100$	$N_{\text{steps}} = 200$	$N_{\text{steps}} = 400$	$N_{\text{steps}} = 800$	$N_{\text{steps}} = 1600$	C
PIRKN	4	1.9/200	3.3/400	5.0/841	6.2/1995	7.3/4800	10^{1}
PISRKN	4	3.0/200	4.6/400	7.0/801	8.2/1601	9.3/3201	10^{1}
PIRKN	6	5.1/360	6.8/800	8.6/1600	10.4/3200	12.2/6400	10^{-1}
PISRKN	6	6.6/246	8.1/443	10.3/809	12.2/1602	14.2/3202	10^{-1}
PIRKN	8	7.7/450	10.1/917	12.5/1934	14.9/4000	17.3/8000	10^{-2}
PISRKN	8	9.8/278	12.2/524	14.5/1002	16.9/1871	19.3/3487	10^{-2}
PIRKN	10	10.4/517	13.3/1050	16.2/2127	19.2/4306	22.2/8706	10^{-2}
PISRKN	10	10.5/314	14.8/558	18.1/1054	22.0/2010		10^{-2}

4.2. Comparison with Sequential Methods

In the section above the PISRKN methods were compared with PIRKN methods (the most efficient parallel methods for nonstiff problems). In this section we will compare the PISRKN methods with the sequential methods currently available.

We restricted our test to the comparison of our tenth-order PISRKN method with a few well-known sequential codes for the orbit problem (4.4). We selected some embedded RKN pairs presented in the form $p(p+1)$ or $(p+1)p$ constructed in [1,2,13,14] and the RKN code DOPRIN taken from [9]. We reproduced the best results obtained by these sequential methods given in the literature (cf., e.g., [7,14]) and added the results obtained by the tenth-order PISRKN method. In spite of the fact the results of the sequential methods are obtained using a stepsize strategy, whereas PISRKN method is applied with fixed stepsizes, it is the PISRKN method that is much more efficient (see Table 6).

Table 6. Comparison with the sequential methods for problem (4.4).

Methods	N_{steps}	NCD	N_{seq}
11(10)-pair (from [14])	919	20.7	15614
8(9)-pair (from [1])	1452	13.5	15973
9(10)-pair (from [2])	628	15.1	8793
	3235	21.4	45291
11(12)-pair (from [13])	876	20.3	17521
DOPRIN (from [9])	1208	12.3	9665
	4466	16.3	35729
	16667	20.3	133337
PISRKN (in this paper)	200	12.4	699
	400	15.4	1244
	800	18.7	2226
	1600	22.3	4295

5. CONCLUDING REMARKS

In this paper, we proposed a special class of s-stage SRKN methods of order $s+1$. The better performance of these SRKN methods is demonstrated when they are used as corrector methods

for generating PC methods for nonstiff problems. By three examples, we have shown that for a given order p, the resulting PISRKN method is by far superior to the PIRKN method (about a factor from 2 to 4). The price we have to pay is a larger number of processors to achieve the same order of accuracy (nearly twice as many). However, comparison of numerical results obtained by a p^{th}-order PISRKN and a p^{th}-order PIRKN method, both implemented on $p/2$ processors (the optimal number of processors for a p^{th}-order PIRKN method), also reveals that the PISRKN method is superior to the PIRKN method.

By comparing the tenth-order PISRKN method with high-order sequential method, we have shown that the PISRKN methods are much more efficient.

REFERENCES

1. E. Fehlberg, Klassische Runge-Kutta-Nyström Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen $x'' = f(t, x)$, *Computing* **10**, 305–315 (1972).
2. E. Fehlberg, Eine Runge-Kutta-Nyström Formel 9-ter Ordnung mit Schrittweitenkontrolle für Differentialgleichungen $x'' = f(t, x)$, *Z. Angew. Math. Mech.* **61**, 477–485 (1981).
3. E. Hairer, Méthodes de Nyström pour l'équations différentielle $y''(t) = f(t, y)$, *Numer. Math.* **27**, 283–300 (1982).
4. E. Hairer, A one-step method of order 10 for $y'' = f(x, y)$, *IMA J. Numer. Anal.* **2**, 83–94 (1982).
5. N.h. Cong, Note on the performance of direct and indirect Runge-Kutta-Nyström methods, *J. Comp. Appl. Math.* **45**, 347–355 (1993).
6. N.h. Cong, Explicit parallel two-step Runge-Kutta-Nyström methods, *Computers Math. Applic.* (to appear).
7. B.P. Sommeijer, Explicit, high-order Runge-Kutta-Nyström methods for parallel computers, *Appl. Numer. Math.* **13**, 221–240 (1993).
8. P.J. van der Houwen, B.P. Sommeijer and N.h. Cong, Stability of collocation-based Runge-Kutta-Nyström methods, *BIT* **31**, 469–481 (1991).
9. E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations, I. Nonstiff Problems*, Springer-Verlag, Berlin, (1987).
10. N.h. Cong, Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial value problems, *J. Comp. Appl. Math.* **51**, 117–125 (1994).
11. P.J. van der Houwen and N.h. Cong, Parallel block predictor-corrector methods of Runge-Kutta type, *Appl. Numer. Math.* **13**, 109–123 (1993).
12. N.h. Cong, A-stable diagonally implicit Runge-Kutta-Nyström methods for parallel computers, *Numerical Algorithms* **4**, 263–281 (1993).
13. S. Filippi and J. Gräf, Eine Runge-Kutta-Nyström Formelpaar der Ordnung 11(12) Differentialgleichungen der Form $y'' = f(t, y)$, *Computing* **34**, 271–282 (1985).
14. S. Filippi and J. Gräf, New Runge-Kutta-Nyström formula-pairs of order 8(7), 9(8) 10(9) and 11(10) for differential equations of the form $y'' = f(t, y)$, *J. Comp. Appl. Math.* **14**, 361–370 (1986).
15. L.F. Shampine and M.K. Gordon, *Computer Solution of Ordinary Differential Equations, The Initial Value Problem*, W.H. Freeman and Company, San Francisco, (1975).
16. M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards Applied Mathematics Series 55, Dover, New York, (1970).
17. E. Fehlberg, S. Filippi and J. Gräf, Eine Runge-Kutta-Nyström Formelpaar der Ordnung 10(11) für Differentialgleichungen $y'' = f(t, y)$, *Z. Angew. Math. Mech.* **66**, 265–270 (1986).
18. W. Glasmacher and D. Sommer, *Implizite Runge-Kutta-Formeln*, Westdeutscher Verlag, Köln, (1966).
19. N.h. Cong, Highly parallel predictor-corrector methods of Runge-Kutta-Nyström type, (in preparation).

APPENDIX

Here we give the paremeters $(\mathbf{c}, A, \mathbf{b}, \mathbf{d})$ in 24 decimal of the corrector RKN methods of fourth-order, sixth-order, eighth-order and tenth-order based on 'optimal' collocation vectors defined in Section 3.2. These parameters were computed on a 28-arithmetic computers.

Table A.1. Parameters $(\mathbf{c}, A, \mathbf{b}, \mathbf{d})$ of the fourth-order corrector RKN method.

$c(1) = 1.05758460000000000000000\text{E}{-01}$	$a(1,3) = 6.00682118829553053088632 8\text{E}{-04}$
$c(2) = 5.00000000000000000000000\text{E}{-01}$	$a(2,3) = -2.57913351693637714245172 9\text{E}{-03}$
$c(3) = 8.94241540000000000000000\text{E}{-01}$	$a(3,3) = 7.19325016909532599534124 3\text{E}{-03}$
$a(1,1) = 7.19325016909532599534124 3\text{E}{-03}$	$b(1) = 2.39728039237067508929829 3\text{E}{-01}$
$a(2,1) = 1.03109035889734129133134 2\text{E}{-01}$	$b(2) = 2.31920260339205994691513 4\text{E}{-01}$
$a(3,1) = 2.11977020932170565604260 5\text{E}{-01}$	$b(3) = 2.83517004237264963786573 7\text{E}{-02}$
$a(1,2) = -2.20150635713907904842987 6\text{E}{-03}$	$d(1) = 2.68079739660794005308486 6\text{E}{-01}$
$a(2,2) = 2.44700976272022480093175 1\text{E}{-02}$	$d(2) = 4.63840520678411989383026 7\text{E}{-01}$
$a(3,2) = 1.80663694829519908400398 2\text{E}{-01}$	$d(3) = 2.68079739660794005308486 6\text{E}{-01}$

Table A.2. Parameters $(\mathbf{c}, A, \mathbf{b}, \mathbf{d})$ of the eighth-order corrector RKN method.

$c(1) = 2.29480800000000000000000\text{E}{-02}$	$a(1,5) = 3.12694273377438423396654 2\text{E}{-05}$
$c(2) = 1.18361190000000000000000\text{E}{-01}$	$a(2,5) = -8.04315398067491232895903 1\text{E}{-05}$
$c(3) = 2.81073520000000000000000\text{E}{-01}$	$a(3,5) = 2.50650678921312046578541 7\text{E}{-04}$
$c(4) = 5.00000000000000000000000\text{E}{-01}$	$a(4,5) = -1.04896624294435981306619 0\text{E}{-03}$
$c(5) = 7.18926480000000000000000\text{E}{-01}$	$a(5,5) = 3.67674371603580244843108 3\text{E}{-03}$
$c(6) = 8.81638810000000000000000\text{E}{-01}$	$a(6,5) = 3.14896090896228741461593 2\text{E}{-02}$
$c(7) = 9.77051920000000000000000\text{E}{-01}$	$a(7,5) = 5.05476118419281701471045 3\text{E}{-02}$
$a(1,1) = 3.54193315498661590858024 2\text{E}{-04}$	$a(1,6) = -1.83728614840803897592256 2\text{E}{-05}$
$a(2,1) = 5.52924640769073897159421 2\text{E}{-03}$	$a(2,6) = 4.47235897253588559911837 2\text{E}{-05}$
$a(3,1) = 1.51352800977220511318905 8\text{E}{-02}$	$a(3,6) = -1.26365231600522208424092 9\text{E}{-04}$
$a(4,1) = 2.79528321502175147962700 0\text{E}{-02}$	$a(4,6) = 4.23355727283452191578685 3\text{E}{-04}$
$a(5,1) = 4.09853692392781507951993 6\text{E}{-02}$	$a(5,6) = -3.12739046115814189793339 7\text{E}{-05}$
$a(6,1) = 5.05077297977970268482636 7\text{E}{-05}$	$a(6,6) = 1.70789178067976106939304 0\text{E}{-03}$
$a(7,1) = 5.61408744618075486954559 7\text{E}{-02}$	$a(7,6) = 1.22426851328332944361159 8\text{E}{-02}$
$a(1,2) = -1.42715204619479118250999 7\text{E}{-04}$	$a(1,7) = 5.76786487360002848286418 3\text{E}{-06}$
$a(2,2) = 1.70789178067976106939303 9\text{E}{-03}$	$a(2,7) = -1.37057540701872514360673 9\text{E}{-05}$
$a(3,2) = 2.10901144128819689955646 7\text{E}{-02}$	$a(3,7) = 3.71830615938281968099898 0\text{E}{-05}$
$a(4,2) = 4.99631877918611075105184 4\text{E}{-02}$	$a(4,7) = -1.14721148249459537216556 3\text{E}{-04}$
$a(5,2) = 7.78319105800612380149114 2\text{E}{-02}$	$a(5,7) = -5.16403215275749366931554 9\text{E}{-05}$
$a(6,2) = 9.91243877188806694938707 0\text{E}{-02}$	$a(6,7) = -8.44246373759955956791565 2\text{E}{-05}$
$a(7,2) = 1.11446691605124003802487 3\text{E}{-01}$	$a(7,7) = 3.54193315498661590858024 2\text{E}{-04}$
$a(1,3) = 8.09495579480480906692595 3\text{E}{-05}$	$b(1) = 5.74852666769887234268415 9\text{E}{-02}$
$a(2,3) = -3.22629981231696923964469 5\text{E}{-04}$	$b(2) = 1.14443912528220301252852 7\text{E}{-01}$
$a(3,3) = 3.67674371603580244843108 3\text{E}{-03}$	$b(3) = 1.40558868870773022372391 4\text{E}{-01}$
$a(4,3) = 4.17538227079646575692799 1\text{E}{-02}$	$b(4) = 1.15844252475943199965373 4\text{E}{-01}$
$a(5,3) = 8.58562285807393468112707 5\text{E}{-02}$	$b(5) = 5.49532909689549876076991 9\text{E}{-02}$
$a(6,3) = 1.17337385432865856711526 4\text{E}{-01}$	$b(6) = 1.53642483990649906149732 0\text{E}{-02}$
$a(7,3) = 1.36103509613135900663467 1\text{E}{-01}$	$b(7) = 1.35016008005477475986847 7\text{E}{-03}$
$a(1,4) = -4.77849117112940443395880 1\text{E}{-05}$	$d(1) = 5.88354267570434981867100 6\text{E}{-02}$
$a(2,4) = 1.39591146120824401711692 8\text{E}{-04}$	$d(2) = 1.29808160927285291867825 9\text{E}{-01}$
$a(3,4) = -5.62444912959240610850774 7\text{E}{-04}$	$d(3) = 1.95512159839728009980090 6\text{E}{-01}$
$a(4,4) = 6.07048901386708728263570 8\text{E}{-03}$	$d(4) = 2.31688504951886399930746 9\text{E}{-01}$
$a(5,4) = 5.01603039326198182858598 8\text{E}{-02}$	$d(5) = 1.95512159839728009980090 6\text{E}{-01}$
$a(6,4) = 8.85609164666378573264660 1\text{E}{-02}$	$d(6) = 1.29808160927285291867825 9\text{E}{-01}$
$a(7,4) = 1.10479661217515620664511 1\text{E}{-01}$	$d(7) = 5.88354267570434981867100 6\text{E}{-02}$

Table A.3. Parameters $(\mathbf{c}, A, \mathbf{b}, \mathbf{d})$ of the tenth-order corrector RKN method.

$c(1) = 1.53245100000000000000000000E-02$	$a(1,6) = -1.78006768507279518029832 6E-05$
$c(2) = 7.95650000000000000000000000E-02$	$a(2,6) = 3.71585203934461835153890 6E-05$
$c(3) = 1.90355530000000000000000000E-01$	$a(3,6) = -7.71164418602790877046436 5E-05$
$c(4) = 3.38246650000000000000000000E-01$	$a(4,6) = 1.49114478782955520316346 2E-04$
$c(5) = 5.00000000000000000000000000E-01$	$a(5,6) = -1.68104734127829224585009 8E-04$
$c(6) = 6.61753350000000000000000000E-01$	$a(6,6) = 2.70380628435801688822364 3E-03$
$c(7) = 8.09644700000000000000000000E-01$	$a(7,6) = 2.32175146877128255328621 7E-02$
$c(8) = 9.20435000000000000000000000E-01$	$a(8,6) = 4.12376465069651543985155 0E-02$
$c(9) = 9.84675490000000000000000000E-01$	$a(9,6) = 5.13294407905645864129365 0E-02$
$a(1,1) = 1.60147589756523730864474 2E-04$	$a(1,7) = 1.10878590290364017210933 6E-05$
$a(2,1) = 2.48638291703082345282281 9E-03$	$a(2,7) = -2.21278775001213652397011 2E-05$
$a(3,1) = 6.87494572039480221294838 1E-03$	$a(3,7) = 4.21022765593286665701799 2E-05$
$a(4,1) = 1.26413538503415166787354 7E-02$	$a(4,7) = -6.65389361422188417508673 6E-05$
$a(5,1) = 1.90169628211235337877527 2E-02$	$a(5,7) = 1.49848648932108924321168 1E-05$
$a(6,1) = 2.53590625003510558397343 2E-02$	$a(6,7) = -3.59581146316027034125805 1E-04$
$a(7,1) = 3.11764478893644296706994 5E-02$	$a(7,7) = 1.76314303070156408796715 6E-03$
$a(8,1) = 3.55148370470695992979594 9E-02$	$a(8,7) = 1.44618940996724562225378 8E-02$
$a(9,1) = 3.80408008439675725192392 8E-02$	$a(9,7) = 2.31692975809105964979779 4E-02$
$a(1,2) = -7.07278616730552329745393 0E-05$	$a(1,8) = -5.85254290180733096241030 2E-06$
$a(2,2) = 7.96483804257794107888250 3E-04$	$a(2,8) = 1.13961366449001180251831 4E-05$
$a(3,2) = 9.69330880887906389057723 5E-03$	$a(3,8) = -2.06993587766399591368851 0E-05$
$a(4,2) = 2.29823916131152952459955 5E-02$	$a(4,8) = 2.93850168785260387178453 6E-05$
$a(5,2) = 3.71590135922274281195882 8E-02$	$a(5,8) = 4.54447135017069728076645 3E-06$
$a(6,2) = 5.14782393525296517804340 8E-02$	$a(6,8) = 1.22307707011906143096762 4E-04$
$a(7,2) = 6.44975142150253366458837 2E-02$	$a(7,8) = -9.74158590734743490171914 1E-05$
$a(8,2) = 7.43203343783994149626402 2E-02$	$a(8,8) = 7.96483804257794107888256 5E-04$
$a(9,2) = 7.99801137817074276170340 7E-02$	$a(9,8) = 5.60630022118166487040689 7E-03$
$a(1,3) = 4.78639475742225851237955 3E-05$	$a(1,9) = 1.72743489538085118196037 4E-06$
$a(2,3) = -1.73431036392925411433530 6E-04$	$a(2,9) = -3.32415851488553543829499 3E-06$
$a(3,3) = 1.76314303070156408796715 7E-03$	$a(3,9) = 5.90606071606236530545830 2E-06$
$a(4,3) = 1.91766971433253501041139 4E-02$	$a(4,9) = -7.96616712022719981030659 7E-06$
$a(5,3) = 4.09187290299824381160839 7E-02$	$a(5,9) = -2.57388341256204627594354 5E-06$
$a(6,3) = 6.22046711043948584673130 9E-02$	$a(6,9) = -3.06908912593919497772294 5E-05$
$a(7,3) = 8.18495906067377831138738 8E-02$	$a(7,9) = 6.41413997097785028504993 7E-06$
$a(8,3) = 9.64206855887437147160354 2E-02$	$a(8,9) = -3.45292864568833818367199 7E-05$
$a(9,3) = 1.04940009822543864761878 9E-01$	$a(9,9) = 1.60147589756523730864471 3E-04$
$a(1,4) = -3.50100587145269433367998 5E-05$	$b(1) = 3.86404347042844386211052 9E-02$
$a(2,4) = 9.13785643724837518456413 1E-05$	$b(2) = 8.13402161696211267746539 2E-02$
$a(3,4) = -3.06255358790765911398098 1E-04$	$b(3) = 1.06953275366291088221931 3E-01$
$a(4,4) = 2.70380628435801688822364 2E-03$	$b(4) = 1.05259420801623568915230 2E-01$
$a(5,4) = 2.55606111593728197876101 5E-02$	$b(5) = 8.12262376843660139185250 0E-02$
$a(6,4) = 5.16065462657842535447066 7E-02$	$b(6) = 5.38019890146222708908398 2E-02$
$a(7,4) = 7.49040853916446103809459 5E-02$	$b(7) = 2.51457870361126337746276 2E-02$
$a(8,4) = 9.26408582499874148545756 0E-02$	$b(8) = 7.03127792786661193003889 1E-03$
$a(9,4) = 1.02804081959429683428860 6E-01$	$b(9) = 6.01361295212246953047963 9E-04$
$a(1,5) = 2.59846122550038901854092 9E-05$	$d(1) = 3.92417959994966855741532 5E-02$
$a(2,5) = -5.86222577915153019857560 7E-05$	$d(2) = 8.83714940974877387046928 3E-02$
$a(3,5) = 1.42279162967313734871214 8E-04$	$d(3) = 1.32099062402403721996558 8E-01$
$a(4,5) = -4.02845165427964434541625 0E-04$	$d(4) = 1.59061409816245839806070 2E-01$
$a(5,5) = 2.49583267859078987011295 3E-03$	$d(5) = 1.62452475368732027837050 0E-01$
$a(6,5) = 2.58743869412569263203944 8E-02$	$d(6) = 1.59061409816245839806070 1E-01$
$a(7,5) = 5.04447897987063970664998 2E-02$	$d(7) = 1.32099062402403721996558 9E-01$
$a(8,5) = 6.82420842238613348216843 8E-02$	$d(8) = 8.83714940974877387046928 3E-02$
$a(9,5) = 7.87627177133081301608012 3E-02$	$d(9) = 3.92417959994966855741532 7E-02$

Table A.4. Parameters $(\mathbf{c}, A, \mathbf{b}, \mathbf{d})$ of the sixth-order corrector RKN method.

$c(1) = 4.28243600000000000000000000\text{E}{-02}$	$a(1,4) = -1.21787386396034810155841 7\text{E}{-04}$
$c(2) = 2.17581710000000000000000000\text{E}{-01}$	$a(2,4) = 3.90679857178352953129770 8\text{E}{-04}$
$c(3) = 5.00000000000000000000000000\text{E}{-01}$	$a(3,4) = -1.59002917878521085396788 1\text{E}{-03}$
$c(4) = 7.82418290000000000000000000\text{E}{-01}$	$a(4,4) = 5.53694097352330775065580 6\text{E}{-03}$
$c(5) = 9.57175640000000000000000000\text{E}{-01}$	$a(5,4) = 4.12264657062557097797594 8\text{E}{-02}$
$a(1,1) = 1.21732485431466886287558 1\text{E}{-03}$	$a(1,5) = 3.78982441942038740463705 6\text{E}{-05}$
$a(2,1) = 1.88423719139440200366130 8\text{E}{-02}$	$a(2,5) = -1.11540157470639787023992 0\text{E}{-04}$
$a(3,1) = 4.98977685677340606551617 6\text{E}{-02}$	$a(3,5) = 3.68623537791582574321640 7\text{E}{-04}$
$a(4,1) = 8.00140216993140415615872 4\text{E}{-02}$	$a(4,5) = -9.03562891562547764559268 9\text{E}{-05}$
$a(5,1) = 9.90961883040791600357266 2\text{E}{-02}$	$a(5,5) = 1.21732485431466886287557 1\text{E}{-03}$
$a(1,2) = -4.53965280428188587005478 3\text{E}{-04}$	$b(1) = 1.03697762839437400077165 3\text{E}{-01}$
$a(2,2) = 5.53694097352330775065580 6\text{E}{-03}$	$b(2) = 1.86610357384477554338389 9\text{E}{-01}$
$a(3,2) = 6.57680307009410458515472 3\text{E}{-02}$	$b(3) = 1.53158169371507560741599 9\text{E}{-01}$
$a(4,2) = 1.35106799616630866364160 0\text{E}{-01}$	$b(4) = 5.18942376250250409273597 1\text{E}{-02}$
$a(5,2) = 1.76274763359740376967639 3\text{E}{-01}$	$b(5) = 4.63947277955244391548510 0\text{E}{-03}$
$a(1,3) = 2.37492473020150660239378 3\text{E}{-04}$	$d(1) = 1.08337235618989843992650 4\text{E}{-01}$
$a(2,3) = -9.87552323912990953374669 2\text{E}{-04}$	$d(2) = 2.38504595009502595265749 7\text{E}{-01}$
$a(3,3) = 1.05556063723185217729372 5\text{E}{-02}$	$d(3) = 3.06316338743015121483199 8\text{E}{-01}$
$a(4,3) = 8.55217842629500891000528 8\text{E}{-02}$	$d(4) = 2.38504595009502595265749 7\text{E}{-01}$
$a(5,3) = 1.40277860680314884353999 0\text{E}{-01}$	$d(5) = 1.08337235618989843992650 4\text{E}{-01}$