ELSEVIER

Applied Numerical Mathematics 31 (1999) 353–374

APPLIED NUMERICAL MATHEMATICS

www.elsevier.nl/locate/apnum

Parallel Störmer–Cowell methods for high-precision orbit computations

P.J. van der Houwen [a,*], E. Messina [b], J.J.B. de Swart [a]

[a] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

[b] *Dip. di Matematica e Applicazioni "R. Caccoppoli", University of Naples "Federico II", Via Cintia, I-80126 Naples, Italy*

Abstract

Many orbit problems in celestial mechanics are described by (nonstiff) initial-value problems (IVPs) for second-order ordinary differential equations of the form $y'' = f(y)$. The most successful integration methods are based on high-order Runge–Kutta–Nyström formulas. However, these methods were designed for *sequential* computer systems. In this paper, we consider high-order *parallel* methods that are not based on Runge–Kutta–Nyström formulas, but which fit into the class of general linear methods. In each step, these methods compute blocks of k approximate solution values (or stage values) at k different points using the *whole* previous block of solution values. The k stage values can be computed in parallel, so that on a k-processor computer system such methods effectively perform as a one-value method. The block methods considered in this paper are such that each equation defining a stage value resembles a linear multistep equation of the familiar Störmer–Cowell type. For $k = 4$ and $k = 5$ we constructed explicit PSC methods with stage order $q = k$ and step point order $p = k + 1$ and implicit PSC methods with $q = k + 1$ and $p = k + 2$. For $k \geqslant 6$ we can construct explicit PSC methods with $q = k$ and $p = k + 2$ and implicit PSC methods with $q = k + 1$ and $p = k + 3$. It turns out that for $k \geqslant 5$ the abscissae of the stage values can be chosen such that only $k - 1$ stage values in each block have to be computed, so that the number of *computational* stages, and hence the number of processors and the number of starting values needed, reduces to $k^* = k - 1$. The numerical examples reported in this paper show that the effective number of right-hand side evaluations required by a variable stepsize implementation of the 10th-order PSC method is 4 up to 30 times less than required by the Runge–Kutta–Nyström code DOPRIN (which is considered as one of the most efficient sequential codes for second-order ODEs). Furthermore, a comparison with the 12th-order parallel code PIRKN reveals that the PSC code is, in spite of its lower order, at least equally efficient, and in most cases more efficient than PIRKN. © 1999 Elsevier Science B.V. and IMACS. All rights reserved.

Keywords: Numerical analysis; General linear methods; Orbit equations; Parallelism

* Corresponding author. E-mail: p.j.van.der.houwen@cwi.nl.

1. Introduction

Many orbit problems in celestial mechanics are described by (nonstiff) initial-value problems (IVPs) for the special second-order ordinary differential equation (ODE)

$$\frac{d^2y}{dt^2} = f(y), \quad y, f \in \mathbb{R}^d, \ t \geqslant t_0, \tag{1.1}$$

where the right-hand side does not contain the derivative of y. The most successful integration methods are based on high-order Runge–Kutta–Nyström formulas. We mention the methods proposed by Dormand and Prince [3], by Fehlberg et al. [4], and by Filippi and Gräf [5,6]. These methods were designed for *sequential* computer systems. The first high-order methods designed for use on a *parallel* computers are the PIRKN (Parallel Iterated Runge–Kutta–Nyström) methods due to Sommeijer [10]. On a parallel computer system these methods are by far superior to the earlier sequential Runge–Kutta–Nyström methods which do not have any scope for parallelism.

In this paper, we consider high-order *parallel* methods that are not based on Runge–Kutta–Nyström formulas, but which fit into the class of general linear methods. In each step, these methods compute blocks of k approximate solution values (or stage values) at k different points using the *whole* previous block of solution values. The k stage values can be computed in parallel, so that on a k-processor computer system such methods *effectively* perform as a one-value method. The block methods considered in this paper are such that each equation defining a stage value resembles a linear multistep equation of the familiar Störmer–Cowell type. Therefore, we shall call these block methods parallel Störmer–Cowell (PSC) methods. They are the second-order-ODE analogue of the parallel Adams methods for first-order ODEs proposed in [11].

For $k = 4$ and $k = 5$ we constructed explicit PSC methods with stage order $q = k$ and step point order $p = k + 1$ and implicit PSC methods with $q = k + 1$ and $p = k + 2$. For $k \geqslant 6$ we can construct explicit PSC methods with $q = k$ and $p = k + 2$ and implicit PSC methods with $q = k + 1$ and $p = k + 3$. It turns out that for $k \geqslant 5$ the abscissae of the stage values can be chosen such that only $k - 1$ stage values in each block have to be computed, so that the number of *computational* stages, and hence the number of processors and the number of starting values needed, reduces to $k^* = k - 1$.

The PSC methods of this paper are uniquely defined by their abscissa vector. Therefore, we spent a lot of attention on the analytical evaluation of the abscissae. For $k = 4$ and $k = 5$, we succeeded in deriving explicit expressions for the abscissae. For $k = 6$ and $k = 7$, the abscissae are partly given explicitly and partly defined by a polynomial equation of degree 4. We shall derive exact rational expressions for the coefficients of these polynomials, so that the user can compute the abscissae and, by means of the abscissae, the other method parameters with any accuracy desired.

An important aspect of block methods is their stability interval. In most block methods, the stability interval is relatively small. Therefore, we computed the stability boundary for all methods derived in this paper. In some cases, the stability interval turns out to be empty. However, slightly relaxing the definition of stability by allowing that the eigenvalues of the stability matrix are in a disk of radius $1 + 10^{-6}$, we obtained acceptably large stability boundaries. For example, the explicit 8-stage PSC method of order 10 has a stability boundary 0.78. Since PSC methods, when run on a parallel computer system, are effectively one-value methods, we can compare this value with the stability boundary of the *scaled* stability boundary of Runge–Kutta–Nyström methods (that is, the stability boundary is divided by the number of right-hand side values per step). For example, the 7th-order Runge–Kutta–Nyström method of Dormand and Prince has a scaled stability boundary 0.32.

The numerical examples reported in this paper contain a two-body orbit problem, the nonlinear Fehlberg problem often used as a stability test problem, and the seven-planet problem PLEI [7] which is considered as a hard test problem for numerical integration techniques (together with a number of other 'real-life' test problems, the PLEI problem can also be found in the CWI Test set for IVP solvers [9]). These examples show that the effective number of right-hand side evaluations required by a variable stepsize implementation of the 10th-order PSC method is 4–30 times less than required by the Runge–Kutta–Nyström code DOPRIN (which is considered as one of the most efficient sequential codes for second-order ODEs). Furthermore, a comparison with the 12th-order parallel code PIRKN reveals that the PSC code is, in spite of its lower order, at least equally efficient, and in most cases more efficient than PIRKN.

2. General linear methods

In 1966 Butcher proposed the general linear method (GLM) formularium in order to describe in a unified way the many first-order-ODE methods available in the literature. Extending this formularium to the second-order ODE (1.1) yields the method

$$Y_{n+1} = (R \otimes I)Y_n + h^2(S \otimes I)F(Y_n) + h^2(T \otimes I)F(Y_{n+1}), \quad n = 0, 1, \ldots. \tag{2.1}$$

Here, I is the k-by-k identity matrix, R, S and T are k-by-k matrices, $\otimes$ denotes the Kronecker product operator, h the stepsize $t_{n+1} - t_n$, and each of the k components $y_{n+1,i}$ of the kd-dimensional solution vector Y_{n+1} represents a numerical approximation to $y(t_n + a_i h)$, to $hy'(t_n + a_i h)$ or "to any other quantity which enables us to construct and describe useful methods" (see [2, p. 339]). The vector $a := (a_i)$ is called the *abscissa vector*, Y_n the *stage vector* and its components y_{ni} the *stage values*. Furthermore, for any vector $Y_n = (y_{ni})$, $F(Y_n)$ contains the right-hand side values $(f(y_{ni}))$. Evidently, we can fix one of the abscissae without loss of generality. We shall put $a_k = 1$.

The GLM (2.1) is completely determined by means of the arrays $\{R, S, T\}$ and the starting vector $Y_0 \approx (y(t_0 + (a_i - 1)h))$ and defines in each step a new block Y_{n+1} of solution values. Thus, given $\{Y_0, R, S, T\}$, (2.1) defines the sequence of block vectors $Y_1, Y_2, \ldots$.

In this paper, we shall assume that all components of Y_{n+1} represent numerical approximations to *solution* values $y(t_n + a_i h)$ and we shall restrict our considerations to the case where T is a *diagonal* matrix with diagonal entries δ_i. Such GLMs will be referred to as GLMs with parallel stages, because all stage values can be computed in parallel. GLMs for second-order ODEs with a full matrix T will be subject of future research.

2.1. Solution of the implicit relations

If the matrix T has one or more nonzero diagonal entries δ_i, then $y_{n+1,i}$ has to be obtained by solving one or more (uncoupled) implicit relations of the form

$$y - \delta_i h^2 f(y) = v_{ni}, \tag{2.2}$$

where v_{ni} represent a d-dimensional vector component of $V_n := (R \otimes I)Y_n + h^2(S \otimes I)F(Y_n)$. Note that Eqs. (2.2) can be solved concurrently, so that only after completion of a full integration step the processors need to exchange their computed results. The conventional way of solving (2.2) in nonstiff situations is a fixed point iteration (briefly FP iteration) process of the form

$$y^{(j)} = \delta_i h^2 f\left(y^{(j-1)}\right) + v_{ni}, \quad j \geqslant 1, \tag{2.3}$$

where $y^{(0)}$ represents an initial stage value iterate. These initial iterates can be generated by the GLM (2.1) with $T = O$ and with the same abscissa vector as the underlying implicit GLM. The process (2.3) satisfies the error recursion

$$y^{(j)} - y = \delta_i h^2 \left(f(y^{(j-1)}) - f(y)\right), \quad j \geqslant 1,$$

leading to the estimate

$$\|y^{(j)} - y\| \leqslant |\delta_i| h^2 \|\partial f / \partial y\| \|y^{(j-1)} - y\|.$$

Hence, the convergence condition becomes

$$h < \frac{1}{\sqrt{\|\delta_i \partial f / \partial y\|}}. \tag{2.4}$$

Thus, we should take care that $|\delta_i|$ is sufficiently small.

2.2. *Consistency*

Consistency is defined by substitution of the exact solution into the GLM and by requiring that the residue vanishes as h tends to zero. The rate by which the residue tends to zero determines the *order of consistency*. We shall call the GLM (and the stage vector Y_{n+1}) *consistent of order* q if the residue upon substitution of the exact solution values $y(t_n + a_i h)$ into (2.1) is of order h^{q+2}. The value of q is often called the *stage order*. The consistency condition leads to a set of order conditions to be satisfied by the matrices R, S and T. In addition, in order to have convergence, the GLM should satisfy the *necessary* condition of *zero-stability*, that is, the matrix R should have its eigenvalues on the unit disk and the eigenvalues of modulus one should have multiplicity not greater than two.

From the consistency definition given above, the order conditions follow immediately. For simplicity of notation, we assume that the ODE is a scalar equation. Using the componentwise definition of functions of vectors, that is, for any function g and vector v, we define $g(v) := (g(v_i))$, we obtain on substitution of the exact solution into (2.1) and expansion in a Taylor series

$$\begin{aligned}
\varepsilon_n &:= RY(t_n) + h^2 SF(Y(t_n)) + h^2 TF(Y(t_{n+1})) - Y(t_{n+1}) \\
&= c(-2)y(t_n) + hc(-1)y^{(1)}(t_n) + \cdots + h^{q+2}c(q)y^{(q+2)}(t_n) + h^{q+3}c(q+1)y^{(q+3)}(t_n) + \cdots, \\
&\quad c(-2) := Re - e, \quad c(-1) := Rb - a, \\
&\quad c(j) := \frac{1}{(j+2)!}\left(Rb^{j+2} - a^{j+2}\right) + \frac{1}{j!}\left(Sb^j + Ta^j\right), \quad j \geqslant 0,
\end{aligned} \tag{2.5}$$

where $b := a - e$, $Y(t_n) = y(t_{n-1} + a_i h)$ denotes the vector containing the exact stage values, and $y^{(j)}(t)$ is the jth derivative of the solution. Hence, requiring the *local error* ε_n to be of order $q + 2$ in h, we conclude that the stage values are (at least) consistent of order q if $c(j) = 0$ for $j = -2, -1, \ldots, q - 1$. The components of $c(q)$ may be considered as the *local* error constants. Although $c(q)$ is the first error vector that does not vanish, it may happen that particular components of $c(q)$ are zero. Thus, particular stage equations may have a higher order of consistency.

In the construction of GLMs, we shall start with a given zero-stable matrix R and a diagonal matrix T with small, nonnegative diagonal entries. The matrix S is then determined by imposing the order conditions. From (2.5) it follows that we obtain stage order $q = k$ if R and S satisfy

$$\begin{aligned} &R\boldsymbol{b}^j = \boldsymbol{a}^j, && j = 0, 1; \\ &R\boldsymbol{b}^j + j(j-1)S\boldsymbol{b}^{j-2} = \boldsymbol{a}^j - j(j-1)T\boldsymbol{a}^{j-2}, && j = 2, \ldots, k+1. \end{aligned} \tag{2.6}$$

Let us introduce the k-by-2 matrices U_x and the k-by-k matrices V_x and W_x:

$$U_x := (\boldsymbol{e}, \boldsymbol{x}), \qquad V_x := (\boldsymbol{x}^2, \ldots, \boldsymbol{x}^{k+1}), \qquad W_x := (2\boldsymbol{e}, 6\boldsymbol{x}, \ldots, k(k+1)\boldsymbol{x}^{k-1}). \tag{2.7}$$

The consistency conditions (2.6) can now be expressed as

$$RU_b = U_a, \qquad SW_b = V_a - RV_b - TW_a. \tag{2.8}$$

Given an abscissa vector $\boldsymbol{a}$ with distinct abscissae, a zero-stable matrix R satisfying the condition $RU_b = U_a$, and any matrix T, we obtain a family of GLMs with stage order $q = k$ by defining

$$S = (V_a - RV_b - TW_a)W_b^{-1}. \tag{2.9}$$

2.3. *Linear stability*

In this paper, the linear stability region $\mathbb{S}$ is defined by the set of points in the complex z-plane where the matrix

$$M(z) := (I - zT)^{-1}(R + zS) \tag{2.10}$$

has its eigenvalues on the unit disk. The process (2.1) will be called linearly stable if the eigenvalues λ of the matrix $h^2\partial f/\partial y$ are in $\mathbb{S}$. Since the problem (1.1) is itself only linearly stable if the eigenvalues of $\partial f/\partial y$ are negative, the intersection of $\mathbb{S}$ with the negative axis is of special interest. If $[-\beta^2, 0]$ is the largest interval contained in $\mathbb{S}$, then β will be called the *stability boundary*.

We should avoid the situation where the matrix $M(h^2\lambda)$ becomes singular, i.e., h^2 should never be equal to $(\delta_i\lambda)^{-1}$. This can only happen if $\delta_i\lambda > 0$. Together with the requirement of linear stability, we are led to the stepsize conditions

$$h < \frac{1}{\sqrt{\max_{\delta_i\lambda>0}\{\delta_i\lambda\}}}, \qquad h < \frac{\beta}{\sqrt{\max_\lambda\{\lambda\}}}. \tag{2.11}$$

Note that the nonsingularity condition is always less restrictive than the convergence condition (2.4) and less restrictive than the stability condition if $\max|\delta_i| < \beta^{-2}$.

3. Parallel Störmer–Cowell methods

In this paper, we restrict our considerations to GLMs where R is given by the zero-stable matrix

$$R = (\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{e} - \boldsymbol{r}, \boldsymbol{r}), \quad r_k = 1 + r_{k-1}. \tag{3.1a}$$

It is easily verified that this matrix has $k-2$ zero eigenvalues and two eigenvalues 1, hence it is zero-stable. On substitution of (3.1a) and $\boldsymbol{b} = \boldsymbol{a} - \boldsymbol{e}$ into $RU_b = U_a$, that is, into $R\boldsymbol{e} = \boldsymbol{e}$, $R\boldsymbol{b} = \boldsymbol{a}$, yields

$$\boldsymbol{r} = \boldsymbol{e} - \frac{\boldsymbol{a}}{a_{k-1} - 1}. \tag{3.1b}$$

Thus, for any abscissae vector $\boldsymbol{a}$, the matrix R defined by (3.1) is zero-stable and it satisfies the consistency condition $RU_b = U_a$. Each stage equation of the GLM $\{(2.1),\ (3.1)\}$ resembles the linear multistep formula of Störmer and Cowell, in the sense that the new y-value is defined using two preceding y-values and k preceding f-values. Therefore, we shall call the GLM $\{(2.1),\ (3.1)\}$ a *parallel Störmer–Cowell* method or briefly PSC method.

3.1. *Order of accuracy at the step points*

If the GLM (2.1) has stage order q, then its local error is given by (cf. (2.5))

$$\varepsilon_n(q) := h^{q+2}\boldsymbol{c}(q)y^{(q+2)}(t_n) + h^{q+3}\boldsymbol{c}(q+1)y^{(q+3)}(t_n) + \cdots, \tag{3.2a}$$

where again, for simplicity of notation, we assume that the ODE is a scalar equation. Let us consider how this local error is propagated by the GLM (2.1). Let α_n denote the global or accumulated error, i.e., $\boldsymbol{Y}_n = \boldsymbol{Y}(t_n) + \alpha_n$. Then

$$\boldsymbol{F}(\boldsymbol{Y}(t_n) + \alpha_n) - \boldsymbol{F}(\boldsymbol{Y}(t_n)) = J_n\alpha_n + \mathrm{O}(\alpha_n^2),$$

where J_n is the k-by-k diagonal matrix whose diagonal entries contain the derivatives of f with respect to y at the points $y(t_{n-1} + a_i h)$. Ignoring second-order terms in α_n, the GLM (2.1) propagates the accumulated error α_n according to

$$\alpha_{n+1} = M_n\alpha_n + D_n\varepsilon_n(q), \quad M_n := \left(I - h^2TJ_{n+1}\right)^{-1}\left(R + h^2SJ_n\right),$$
$$D_n := \left(I - h^2TJ_{n+1}\right)^{-1}. \tag{3.2b}$$

Note that for the scalar test equation $y' = \lambda y$, the matrix M_n reduces to the matrix $M(h^2\lambda)$, where $M(\cdot)$ is the stability amplification matrix defined in (2.10). Suppose that the vector $\boldsymbol{Y}_{n-s+1}$ is exact at t_{n-s+1}, i.e., $\alpha_{n-s+1} = \boldsymbol{0}$. Then, the accumulated error over s steps is given by

$$\alpha_{n+1} = D_n\varepsilon_n(q) + M_nD_{n-1}\varepsilon_{n-1}(q) + M_nM_{n-1}D_{n-2}\varepsilon_{n-2}(q) + \cdots$$
$$+ M_n\cdots M_{n-s+2}D_{n-s+1}\varepsilon_{n-s+1}(q). \tag{3.2'}$$

We are particularly interested in the accumulated *step point* errors $\alpha_{n+1,k}$. Since $M_n = R + \mathrm{O}(h^2)$, $M_nM_{n-1} = R^2 + \mathrm{O}(h^2), \ldots$, and because for PSC methods the first $k-2$ columns of R^j vanish, only the last two components of $D_j\varepsilon_j(q)$ play a role in the $\mathrm{O}(h^{q+2})$ and $\mathrm{O}(h^{q+3})$ terms of $\alpha_{n+1,k}$. In fact, $\alpha_{n+1,k} = \mathrm{O}(h^{q+3})$ if $c_{k-1}(q) = c_k(q) = 0$ and $\alpha_{n+1,k} = \mathrm{O}(h^{q+4})$ if $c_{k-1}(q+1) = c_k(q+1) = 0$. Thus, in PSC methods, we can achieve step point order $p = q+1$ and $p = q+2$ (superconvergence) by imposing conditions on the last two components of the error vectors $\boldsymbol{c}(q)$ and $\boldsymbol{c}(q+1)$, respectively.

In the following, we distinguish between *explicit* PSC methods ($T = O$) and *implicit* PSC methods ($T \neq O$). In the case of *explicit* PSC methods, the stage order $q = k$, but as shown above, the step point order becomes $p = k+1$ if the abscissae can be chosen such that

$$c_{k-1}(k) = c_k(k) = 0 \tag{3.3a}$$

and it can be raised to $p = k+2$ if in addition to (3.3a),

$$c_{k-1}(k+1) = c_k(k+1) = 0. \tag{3.3b}$$

If the method is *implicit*, then we first use the matrix T to obtain stage order $q = k+1$. Given $\boldsymbol{a}$, R, T and defining S by (2.9), the error vector $\boldsymbol{c}(k)$ can be written as

$$c(k) := \frac{1}{(k+2)!}\big(T\boldsymbol{m}(k) - \boldsymbol{n}(k)\big), \quad \boldsymbol{m}(k) := (k+1)(k+2)\big(\boldsymbol{a}^k - W_a W_b^{-1}\boldsymbol{b}^k\big),$$
$$\boldsymbol{n}(k) := \boldsymbol{a}^{k+2} - R\boldsymbol{b}^{k+2} - (k+1)(k+2)\big((V_a - RV_b)W_b^{-1}\boldsymbol{b}^k\big). \tag{3.4}$$

Hence, the stage order can be raised to $q = k+1$ by setting $\boldsymbol{c}(k) = \boldsymbol{0}$, that is, $T\boldsymbol{m}(k) = \boldsymbol{n}(k)$. Let the abscissae vector $\boldsymbol{a}$ be such that $\boldsymbol{m}(k)$ has nonzero entries. Then all stage values in the GLM (2.1) have order of consistency $q = k+1$ if the matrices S and T are defined by

$$T = \mathrm{diag}\big(\boldsymbol{n}(k)\boldsymbol{m}^{-1}(k)\big), \qquad S = (V_a - RV_b - TW_a)W_b^{-1}. \tag{3.5}$$

We remark that the ith diagonal entry of T can be chosen arbitrary whenever $m_i(k)$ and $n_i(k)$ both vanish. Furthermore, if $m_i(k) = 0$ and $n_i(k) \neq 0$, then the stage order cannot be raised to $k+1$, unless the ith row of S vanishes. Assuming that $m_i(k) = 0$ implies $n_i(k) = 0$, the step point order is raised to $p = k+2$ if the abscissae can be chosen such that

$$c_{k-1}(k+1) = c_k(k+1) = 0, \tag{3.6a}$$

and to $p = k+3$ if, in addition,

$$c_{k-1}(k+2) = c_k(k+2) = 0. \tag{3.6b}$$

One option for the derivation of superconvergent abscissae a_i, is to solve the (highly nonlinear) conditions (3.3) and (3.6). However, it turns out that using the theory of quadrature formulas yields more simple conditions for generating superconvergent abscissae.

3.2. *Conditions for superconvergent abscissae using quadrature formulas*

In the PSC case $\{(2.1), (3.1)\}$, the stage equations can be written as

$$y_{n+a_i} - (1-r_i)y_{n-1+a_{k-1}} - r_i y_{n-1+a_k} = h^2(e_i^{\mathrm{T}} S \otimes I)F(Y_n) + h^2\delta_i f(y_{n+a_i}),$$
$$i = 1, \ldots, k, \tag{3.7}$$

where $y_{n+\theta}$ corresponds with the numerical approximation at the point $t_{n+\theta} := t_n + \theta h$. We compare these equations with their analytical analogue. The main tool is Taylor's theorem with remainder term which yields for the solution of (1.1) the formula

$$\begin{aligned} y(t+\Delta) &= y(t) + \Delta y'(t) + \int_t^{t+\Delta} (t+\Delta-x)y''(x)\,\mathrm{d}x \\ &= y(t) + \Delta y'(t) + \int_t^{t+\Delta} (t+\Delta-x)f\big(y(x)\big)\,\mathrm{d}x. \end{aligned}$$

It can now be verified that the solution of (1.1) satisfies the relations

$$\begin{aligned} &y(t_{n+a_i}) - (1-r_i)y(t_{n-1+a_{k-1}}) - r_i y(t_n) = a_i^2h^2 \int_0^1 g_i(\xi) f\big(y(t_n + a_i h\xi)\big)\,\mathrm{d}\xi, \\ &g_i(\xi) = \begin{cases} 1 - \xi - (1-r_i)(b_{k-1}a_i^{-1} - \xi) & \text{for } 0 \leqslant \xi \leqslant b_{k-1}a_i^{-1}, \\ 1 - \xi & \text{for } b_{k-1}a_i^{-1} \leqslant \xi \leqslant 1, \end{cases} \end{aligned} \tag{3.8}$$

and where we assumed that $0 < b_{k-1} \leqslant 1 + b_i$ (this implies that $a_i \neq 0$). If this constraint is not satisfied, then the function g_i has to be redefined, but it turns out that the constraint on the b_i does not limit us in the construction of useful PSC methods.

Next, we approximate the integral terms in (3.8) by a quadrature formula. Since the functions g_i are not differentiable, we use a formula based on *product integration.* Using the points $b_j a_i^{-1}$ (and 1 in the case of implicit PSC methods) as quadrature points, we may write

$$\int_0^1 g_i(\xi) f\big(y(t_n + a_i h\xi)\big)\,\mathrm{d}\xi = w_{i1} f\big(y(t_n + hb_1)\big) + w_{i2} f\big(y(t_n + hb_2)\big) + \cdots, \tag{3.9}$$

where the w_{ij} are the corresponding quadrature weights which take the function g_i into account. From (3.7)–(3.9) it follows that the right-hand sides in (3.7) may be interpreted as quadrature formulas for the integral terms in (3.8). Let us define $b_{k+1} a_i^{-1} = 1$ for all i. Then, the quadrature points are given by the k points $\{b_j a_i^{-1},\ j = 1, \dots, k\}$ if the PSC method is explicit and by the $k + 1$ points $\{b_j a_i^{-1},\ j = 1, \dots, k + 1\}$ if the PSC method is implicit.

In the following formulas, we introduce the parameter τ to indicate explicit PSC methods ($\tau = 0$) and implicit PSC methods ($\tau = 1$). It can be shown that the quadrature weights $w_{ij}(\tau)$ corresponding with the quadrature points $b_j a_i^{-1}$ are given by (see, e.g., [1, p. 886])

$$w_{ij}(\tau) = a_i^2 h^2 \int_0^1 g_i(\xi) L_{ij}(\tau, \xi)\,\mathrm{d}\xi, \qquad L_{ij}(\tau, \xi) := \prod_{r=1,\ r\neq j}^{k+\tau} \frac{a_i \xi - b_r}{b_j - b_r}.$$

Hence, $h^2 e_i^{\mathrm{T}} S e_j = w_{ij}$ for $j = 1, \dots, k$ and $h^2 \delta_i = \tau w_{i,k+1}$, so that the matrices S and T are given by

$$S = \big(e_i^{\mathrm{T}} S e_j\big) = \left(a_i^2 \int_0^1 g_i(\xi) L_{ij}(\tau, \xi)\,\mathrm{d}\xi \right), \qquad T = \operatorname{diag}\left(\tau a_i^2 \int_0^1 g_i(\xi) L_{i,k+1}(\tau, \xi)\,\mathrm{d}\xi \right).$$

These expressions for the matrices S and T are equivalent with the expressions derived in the preceding section, but have the advantage that the entries are explicitly expressed in terms of the abscissae. For example, it can now immediately seen that the ith row of S and T vanish if g_i is identically zero. This happens if, and only if, $a_{k-1} = \frac{3}{2}$ and $a_i = \frac{1}{2}$ for some $i \leqslant k - 2$, say for $i = k - 2$. PSC methods with this property are attractive from a computational point of view because the corresponding stage equation reduces to $y_{n+a_{k-2}} = y_{n-1+a_{k-1}}$, that is, $y_{n+1,k-2} = y_{n,k-1}$. Hence, the $(k - 2)$nd component of $F(Y_{n+1})$ equals the $(k - 1)$st component of $F(Y_n)$, so that the $(k - 2)$nd processor is not really needed, that is, computationally the PSC method has only $k - 1$ stages. Thus, we have:

Theorem 3.1. *Let all b_i be real and distinct, satisfying the constraint $0 < b_{k-1} \leqslant 1 + b_i$, $i = 1, \dots, k$. Then, the PSC method has $k^* = k - 1$ computational stages if $b_{k-1} = \frac{1}{2}$ and if $b_{k-2} = -\frac{1}{2}$.*

The quadrature error of the product integration formula (3.9) is given by

$$Q_i = \frac{1}{(k+\tau)!} a_i^2 h^2 \int_0^1 p_i(\tau, \xi) \phi_i\big(\tau, \theta(\xi)\big)\,\mathrm{d}\xi, \qquad p_i(\tau, \xi) := g_i(\xi)(\xi - 1)^\tau \prod_{j=1}^{k} \big(\xi - b_j a_i^{-1}\big),$$

$$\phi_i(\tau, \theta) := \frac{\mathrm{d}^{k+\tau} f\big(y(t_n + a_i h\theta)\big)}{\mathrm{d}\theta^{k+\tau}} = (a_i h)^{k+\tau} \frac{\mathrm{d}^{k+\tau} f\big(y(t_n + a_i h\theta)\big)}{\mathrm{d}(a_i h\theta)^{k+\tau}},$$

where $\theta(\xi)$ assumes values in the interval $[0,1]$. We remark that this error formula holds for any integrable function $g(\xi)$ and sufficiently differentiable function $f(y(t))$. By writing

$$Q_i(\tau) = \frac{a_i^2 h^2}{(k+\tau)!} \int_0^1 p_i(\tau,\xi)\left\{\phi_i(\tau,\theta(0)) + \xi\frac{\partial\phi_i(\tau,\theta(0))}{\partial\xi} + \frac{1}{2}\xi^2\frac{\partial^2\phi_i(\tau,\theta(0))}{\partial\xi^2} + \cdots\right\} \mathrm{d}\xi,$$

and observing that each differentiation of the function $\phi_i(\tau,\theta(\xi))$ with respect to ξ increases its order with respect to h, we see that the order of the quadrature error $Q_i(\tau)$ increases by one if the quadrature points $\{b_j a_i^{-1}\colon j = 1,\ldots,k\}$ satisfy the condition

$$\int_0^1 p_i(\tau,\xi)\,\mathrm{d}\xi = 0, \tag{3.10a}$$

and by two if, in addition,

$$\int_0^1 \xi p_i(\tau,\xi)\,\mathrm{d}\xi = 0. \tag{3.10b}$$

From the structure of the matrix R and the factor h^2 in front of the derivative terms in the PSC method it follows that these conditions need only to be satisfied for $i = k-1, k$ (compare our discussion in Section 3.1). The following theorem summarizes the above result.

Theorem 3.2. *Let all b_i be real and distinct, satisfying the constraint $0 < b_{k-1} \leqslant 1 + b_i$, $i = 1,\ldots,k$. Then, the PSC method has step point order*
(a) $p = k + 1 + \tau$ *if* (3.10a) *is satisfied for $i = k-1, k$,*
(b) $p = k + 2 + \tau$ *if both* (3.10a) *and* (3.10b) *are satisfied for $i = k-1, k$.*

Let us apply this theorem for $k = 3$. Solving (3.10a) with $i = 2, 3$ using Maple yields an explicit fourth-order method ($\tau = 0$) defined by

$$b_1 = \tfrac{1}{2}\sqrt{22 - 2\sqrt{109}} \approx 0.529005, \qquad b_2 = 1 + \tfrac{7}{10}b_1 - \tfrac{1}{10}(1+b_1)^2 = 1.136518 \tag{3.11a}$$

and an implicit fifth-order method ($\tau = 1$) defined by

$$b_1 = -\tfrac{1}{10}\sqrt{130 - 10\sqrt{129}} \approx -0.405238, \qquad b_2 = \tfrac{13}{10} + \tfrac{3}{2}b_1 - \tfrac{1}{2}(1+b_1)^2 = 0.515271. \tag{3.11b}$$

For $k = 4$, Maple offers already problems to find an analytical solution. A numerical approach yields an explicit fifth-order method defined by

$$b_1 \approx 1.083424930, \qquad b_2 \approx 2.283072685, \qquad b_3 \approx 0.494330785. \tag{3.12}$$

For $k \geqslant 5$ we have to solve the four equations (3.10) with $i = k-1, k$ simultaneously. This direct approach becomes quite cumbersome even when using numerical techniques, unless we can guess an accurate initial approximation to the solution. An alternative is to assume that the abscissa a_{k-1} is given in advance. Then it is possible to derive an analytically given polynomial equation whose solutions define superconvergent abscissae.

3.3. Superconvergent PSC methods

From now on, we assume that a_{k-1} is prescribed. Let $k \geqslant 4$ and let the abscissae a_j be given for $j \geqslant 3$. Then we can write

$$p_i(\tau,\xi) = (\xi - b_1 a_i^{-1})(\xi - b_2 a_i^{-1}) q_i(\tau,\xi), \qquad q_i(\tau,\xi) := g_i(\xi)(\xi - 1)^\tau \prod_{j=3}^{k} (\xi - b_j a_i^{-1}),$$

where $q_i(\tau,\xi)$ does not contain unknown parameters. Similarly, for $k \geqslant 6$ we assume that the abscissae a_j are given for $j \geqslant 5$ and we write

$$p_i(\tau,\xi) = \prod_{j=1}^{4} (\xi - b_j a_i^{-1}) r_i(\tau,\xi), \qquad r_i(\tau,\xi) := g_i(\xi)(\xi - 1)^\tau \prod_{j=5}^{k} (\xi - b_j a_i^{-1}).$$

Furthermore, we define the integrals

$$I_{ij}(t) := \int_0^1 (a_i\xi)^j q_i(\tau,\xi)\,\mathrm{d}\xi, \qquad J_{ij}(t) := \int_0^1 (a_i\xi)^j r_i(\tau,\xi)\,\mathrm{d}\xi. \tag{3.13}$$

Then, condition (3.10a) holds if the shifted abscissae b_1 and b_2 satisfy the two equations

$$A_1 := b_1 + b_2, \qquad A_2 := b_1 b_2,$$

where A_1 and A_2 are the solution of the two linear equations

$$I_{i1}(\tau)A_1 - I_{i0}(\tau)A_2 = I_{i2}(\tau), \quad i = k-1, k, \ k \geqslant 4. \tag{3.14a}$$

The coefficients $I_{i1}(\tau)$, $I_{i0}(\tau)$ and $I_{i2}(\tau)$ do not depend on b_1 and b_2, so that they are completely determined. Having computed the quantities A_1 and A_2, the abscissae b_1 and b_2 are defined as the roots of the equation $b^2 - A_1 b + A_2 = 0$.

Likewise, (3.10a) and (3.10b) are both satisfied if b_1, b_2, b_3, and b_4 satisfy the equations

$$B_1 := b_1 + b_2 + b_3 + b_4, \qquad B_2 := b_1b_2 + b_1b_3 + b_1b_4 + b_2b_3 + b_2b_4 + b_3b_4,$$

$$B_3 := b_1b_2b_3 + b_1b_2b_4 + b_1b_3b_4 + b_2b_3b_4, \qquad B_4 := b_1b_2b_3b_4,$$

where B_1, B_2, B_3, and B_4 are the solution of the four linear equations

$$\begin{aligned} J_{i3}(\tau)B_1 - J_{i2}(\tau)B_2 + J_{i1}(\tau)B_3 - J_{i0}(\tau)B_4 &= J_{i4}(\tau), \\ J_{i4}(\tau)B_1 - J_{i3}(\tau)B_2 + J_{i2}(\tau)B_3 - J_{i1}(\tau)B_4 &= J_{i5}(\tau), \end{aligned} \quad i = k-1, k, \ k \geqslant 6, \tag{3.14b}$$

in which the coefficients do not depend on b_1, b_2, b_3, and b_4. The parameters b_1, b_2, b_3, and b_4 are now defined as the roots of the equation $b^4 - B_1b^3 + B_2b^2 - B_3b + B_4 = 0$.

We recall that in the derivations above, it is assumed that all a_i are distinct and that the constraint $0 < b_{k-1} \leqslant 1 + b_i$ is satisfied for $i = 1, \ldots, k$. Furthermore, we observe that using formula manipulation software enables us to find exact values for the quantities A_i and B_i, so that the *superconvergence* equations $b^2 - A_1 b + A_2 = 0$ and $b^4 - B_1 b^3 + B_2 b^2 - B_3 b + B_4 = 0$ are obtained in analytical form. The following theorem summarizes the preceding considerations:

Theorem 3.3. *Let the quantities A_i and B_i be defined by* (3.14a) *and* (3.14b), *let all b_i be real and distinct satisfying the constraint $0 < b_{k-1} \leqslant 1 + b_i$ for $i = 1, \ldots, k$. Then, the PSC method has step point order*

(a) $p = k + 1 + \tau$ *if b_1 and b_2 satisfy* $b^2 - A_1 b + A_2 = 0$,
(b) $p = k + 2 + \tau$ *if b_1, b_2, b_3, and b_4 satisfy* $b^4 - B_1 b^3 + B_2 b^2 - B_3 b + B_4 = 0$.

By means of this theorem the polynomial equation to be satisfied by the abscissae of superconvergent PSC methods can straightforwardly be constructed, provided that the abscissa $a_{k-1} = b_{k-1} + 1$ is prescribed in advance. In the following, we shall always choose $b_{k-1} = \frac{1}{2}$. This choice is motivated by the following observations: (i) the methods defined by (3.11) and (3.12) show that one of the abscissae b_i seems always to be close to $\frac{1}{2}$, and (ii) if the method contains sufficiently many free parameters, then we may set $b_{k-2} = -\frac{1}{2}$, so that by virtue of Theorem 3.1 the number of computational stages is reduced to $k - 1$.

In Sections 3.4 and 3.5, we derive with the help of Maple the superconvergence equations $b^2 - A_1 b + A_2 = 0$ and $b^4 - B_1 b^3 + B_2 b^2 - B_3 b + B_4 = 0$ in analytical form for $k = 4, \ldots, 8$ and $\tau = 0, 1$. The resulting (shifted) abscissa vector $\boldsymbol{b}$ (or approximations to it) are listed in Table 1. Since, in principle, we can associate with each abscissa vector both an explicit PSC method (predictor method) defined by $\{(2.9),\ T = O\}$ and an implicit PSC method (corrector method) defined by (3.5), we have also listed the orders of accuracy $\boldsymbol{p} := (p_{\text{pred}}, p_{\text{corr}})$ of the PC pairs (for a derivation of these orders of accuracy, we refer to Sections 3.4 and 3.5). Finally, we listed the number of computational stages k^* (see Theorem 3.1).

3.4. PSC methods with $k \leqslant 5$

First we construct PSC methods with at most 5 stages using part (a) of Theorem 3.1. If the PSC method is explicit ($\tau = 0$) and has 4 stages, then the method is uniquely defined by the equation

$$b^2 - \tfrac{37}{10} b + \tfrac{57}{20} = 0, \qquad b_3 = \tfrac{1}{2}. \tag{3.15}$$

Thus, (3.15) generates an explicit PSC method of order $p = 5$ and an implicit PSC method of order 5, both with 4 parallel stages. The solutions of (3.15) can be found in Table 1 and turn out to be close to the solutions (3.12) where we did not fix the abscissae b_3 in advance.

For $k = 5$ we can construct a one-parameter family of methods of order $p = 6$. Given the abscissa b_3, the abscissae b_1 and b_2 follow from the superconvergence condition

$$b^2 - \frac{74b_3^2 - 195b_3 + 124}{20b_3^2 - 74b_3 + 57} b + \frac{114b_3^2 - 248b_3 + 131}{2(20b_3^2 - 74b_3 + 57)} = 0, \qquad b_4 = \tfrac{1}{2}. \tag{3.16}$$

The free parameter b_3 can be exploited by setting $b_3 = -\frac{1}{2}$, so that we have only 4 computational stages (see Theorem 3.1).

In a similar way, we find the implicit 4-stage PSC method of order 6 defined by

$$b^2 - b + \tfrac{1}{40} = 0, \qquad b_3 = \tfrac{1}{2}, \tag{3.17}$$

and the implicit 5-stage PSC method of order 7 with 4 computational stages defined by

$$b^2 - \tfrac{445}{812} b + \tfrac{1231}{2436} = 0, \qquad b_3 = -\tfrac{1}{2}, \qquad b_4 = \tfrac{1}{2}. \tag{3.18}$$

Table 1
Abscissa vector $\boldsymbol{b}$ for PSC methods. Solutions of the superconvergence conditions

k	k^*	p	Abscissa Eq.	$\boldsymbol{b}=\boldsymbol{a}-\boldsymbol{e}$
4	4	$(5,5)$	(3.15)	$\left(\frac{37+\sqrt{229}}{20}\ \frac{37-\sqrt{229}}{20}\ \frac{1}{2}\ 0\right)$
5	4	$(6,6)$	(3.16)	$\left(\frac{80-\sqrt{163}}{66}\ \frac{80+\sqrt{163}}{66}\ -\frac{1}{2}\ \frac{1}{2}\ 0\right)$
4	4	$(4,6)$	(3.17)	$\left(\frac{10+\sqrt{110}}{20}\ \frac{10-\sqrt{110}}{20}\ \frac{1}{2}\ 0\right)$
5	4	$(5,7)$	(3.18)	$\left(\frac{1335-\sqrt{13777089}}{4872}\ \frac{1335+\sqrt{13777089}}{4872}\ -\frac{1}{2}\ \frac{1}{2}\ 0\right)$
6	6	$(8,8)$	(3.19)	$(0.220473884991749550773176296\ 0.785748179438222426650898115$ $1.082801901339905567884428919\ 1.357404605658693883262925242$ $\frac{1}{2}\ 0)$
7	6	$(9,9)$	(3.20)	$(1.359849808362845524482247436\ 1.085502432861554845592192032\ 38$ $0.783141526651761362293102021\ 9\ 0.223660672730360134033723069\ 79$ $-\frac{1}{2}\ \frac{1}{2}\ 0)$
8	7	$(10,10)$	(3.21)	$(1.347691904907298754183065141\ 60\ 1.072080312447516818672381997\ 82$ $0.786086152017853260021754689\ 95\ 0.225168248342102287044467884\ 14$ $\frac{39}{20}\ -\frac{1}{2}\ \frac{1}{2}\ 0)$
6	6	$(6,9)$	(3.22)	$(0.217555802077306973293458693\ 75\ 0.802119535995225835181126479\ 52$ $1.097331887383193846395423935\ 92\ 1.348784066873229806774773199\ 02$ $\frac{1}{2}\ 0)$
7	6	$(7,10)$	(3.23)	$(1.310559256072037540030205712\ 95\ 1.050304685820485007384623000\ 65$ $0.776141401474259294795865976\ 93\ 0.226170110066294406256058005\ 7$ $-\frac{1}{2}\ \frac{1}{2}\ 0)$
8	7	$(8,11)$	(3.24)	$(1.329260387472804075727248311\ 06\ 1.076174740828738092849623879\ 98$ $0.791207326331779803313801850\ 23\ 0.223056528893693765291593409\ 00$ $\frac{37}{20}\ -\frac{1}{2}\ \frac{1}{2}\ 0)$

3.5. *PSC methods with $k \geqslant 6$*

For $k \geqslant 6$ we can invoke part (b) of Theorem 3.1. Three types of PSC methods will be considered:
(I) Explicit PSC methods of order $k+2$.

(II) Implicit PSC methods of order $k+3$.
(III) Predictor-corrector pairs of PSC methods of order $(k+2, k+2)$.

3.5.1. *Explicit PSC methods of order* $k+2$

For $k=6$ we have a uniquely defined method of order $p=8$ defined by

$$b^4 - \tfrac{193}{56}b^3 + \tfrac{19279}{4704}b^2 - \tfrac{17891}{9408}b + \tfrac{1597}{6272} = 0, \qquad b_5 = \tfrac{1}{2}. \tag{3.19}$$

For $k \geqslant 7$ we obtain methods of order $p = k+2$ with $k-6$ free parameters $b_5, \ldots, b_{k-3}$. For $k=7$ we used the free parameter b_5 to reduce the number of computational stages to 6 and found the equation

$$b^4 - \tfrac{235865}{68324}b^3 + \tfrac{210776}{51243}b^2 - \tfrac{3139325}{1639776}b + \tfrac{423971}{1639776} = 0, \qquad b_5 = -\tfrac{1}{2}, \qquad b_6 = \tfrac{1}{2}. \tag{3.20}$$

Finally, for $k=8$ we have two free parameters. We set $b_6 = -\frac{1}{2}$ and used b_5 to reduce the size of the error constant $E(k+2)$ according to the approach described in Section 3.6, to obtain

$$\begin{aligned} &b^4 - \tfrac{16493095751}{4814898736}b^3 + \tfrac{117118655069}{28889392416}b^2 - \tfrac{217047351761}{115557569664}b + \tfrac{88026108193}{346672708992} = 0, \\ &b_5 = \tfrac{39}{20}, \qquad b_6 = -\tfrac{1}{2}, \qquad b_7 = \tfrac{1}{2}. \end{aligned} \tag{3.21}$$

3.5.2. *Implicit PSC methods of order* $k+3$

Proceeding as in the previous section, we solve Eqs. (3.14b) for $\tau = 1$, to obtain a 6-stage, 9th-order method defined by

$$b^4 - \tfrac{5015}{1447}b^3 + \tfrac{18010}{4341}b^2 - \tfrac{67235}{34728}b + \tfrac{251147}{972384} = 0, \qquad b_5 = \tfrac{1}{2}, \tag{3.22}$$

a 7-stage, 10th-order method with 6 computational stages

$$b^4 - \tfrac{9023504}{2683031}b^3 + \tfrac{157695722}{40245465}b^2 - \tfrac{14440832}{8049093}b + \tfrac{71811311}{297197280} = 0, \qquad b_5 = -\tfrac{1}{2}, \qquad b_6 = \tfrac{1}{2}, \tag{3.23}$$

and an 8-stage, 11th-order method with 7 computational stages whose free parameter b_5 was used to minimize the error constant $E(k+3)$:

$$\begin{aligned} &b^4 - \tfrac{109326306018669}{31969569995869}b^3 + \tfrac{1293727397185447}{319695699958690}b^2 - \tfrac{479656555759929}{255756559966952}b + \tfrac{387414729958955 9}{1534539359801712 0} = 0, \\ &b_5 = \tfrac{37}{20}, \qquad b_6 = -\tfrac{1}{2}, \qquad b_7 = \tfrac{1}{2}. \end{aligned} \tag{3.24}$$

3.5.3. *PC pairs of order* $(k+2, k+2)$

The explicit methods of type I and the implicit methods of type II constructed in the two preceding sections possess an optimal order of accuracy $k+2$ and $k+3$, respectively. A third option is to construct a PC pair with the *same* abscissae vector such that the predictor is of order $k+1$ and the corrector of order $k+2$. This can be achieved by determining a single shifted-abscissae vector $\boldsymbol{b}$ such that (3.10a) is satisfied both for $\tau = 0$ and for $\tau = 1$. Proceeding as in Section 3.2, we first solve the four linear equations

$$J_{i3}(\tau)B_1 - J_{i2}(\tau)B_2 + J_{i1}(\tau)B_3 - J_{i0}(\tau)B_4 = J_{i4}(\tau), \qquad i = k-1, k, \; k \geqslant 6, \quad \tau = 0, 1, \tag{3.25}$$

where $J_{ij}(\tau)$ is defined as in (3.13). The abscissae b_1, b_2, b_3, and b_4 are again defined as the roots of the superconvergence condition $b^4 - B_1 b^3 + B_2 b^2 - B_3 b + B_4 = 0$. Thus, if this equation has real, distinct roots, then the resulting PC pair has order $(k+1, k+2)$. However, it turns out that the actual order is $(k+2, k+2)$.

Theorem 3.4. *Let the quantities B_i be defined by* (3.25), *let all b_i be real and distinct such that $0 < b_{k-1} \leqslant 1 + b_i$ for $i = 1, \ldots, k$, and let b_1, b_2, b_3, b_4 satisfy $b^4 - B_1 b^3 + B_2 b^2 - B_3 b + B_4 = 0$. Then, the abscissae vector $\boldsymbol{b}$ generates an explicit PSC method and an implicit PSC method which have both of step point order $p = k+2$.*

Proof. The proof consists of showing that Eqs. (3.25) are identical with Eqs. (3.14b) obtained for $\tau = 0$, i.e., with the equations

$$\begin{aligned} &J_{i3}(0)B_1 - J_{i2}(0)B_2 + J_{i1}(0)B_3 - J_{i0}(0)B_4 = J_{i4}(0), \\ &J_{i4}(0)B_1 - J_{i3}(0)B_2 + J_{i2}(0)B_3 - J_{i1}(0)B_4 = J_{i5}(0), \end{aligned} \quad i = k-1, k,\ k \geqslant 6. \tag{3.26}$$

Evidently, the first two equations in (3.26) are identical with the two equations in (3.25) obtained for $\tau = 0$. Furthermore, by observing that $J_{ij}(\tau)$ satisfies the relation $J_{i,j+1}(0) = a_i(J_{ij}(1) + J_{ij}(0))$, it follows that the last two equations in (3.26) can be written as

$$\begin{aligned} &\big(J_{i3}(0) + J_{i3}(1)\big)B_1 - \big(J_{i2}(0) + J_{i2}(1)\big)B_2 + \big(J_{i1}(0) + J_{i1}(1)\big)B_3 - \big(J_{i0}(0) + J_{i0}(1)\big)B_4 \\ &\quad = J_{i4}(0) + J_{i4}(1), \quad i = k-1, k, \end{aligned}$$

which is by virtue of the first two equations of (3.26) identical with the two equations in (3.25) obtained for $\tau = 1$. This proves the assertion of the theorem. □

Theorem 3.4 implies that for $k \geqslant 6$ the abscissae vectors (3.19)–(3.21) derived in Section 3.4 not only generate predictors of order $k+2$, but also correctors of order $k+2$.

3.6. Comparison of PSC methods

In order to compare the accuracy of the various PSC methods we again consider the formula (3.2′) for the accumulated error α_{n+1}. The conventional approach is to compare the local errors $\varepsilon_j(q)$ by means of the error constants $c(q)$, $c(q+1), \ldots$ given in (3.2a). However, then the amplifying effect of the matrix M_n is not taken into account. Therefore, we consider the accumulated error after s steps. On substitution of $\varepsilon_j(q)$ into (3.2′) and writing $J_{n+1} = (\partial f(y(t_n))/\partial y)I + \mathrm{O}(h)$ in the definition of M_n and D_n, we can easily find the first few terms of the expansion of the step point value of α_{n+1}.

Theorem 3.5. *After s steps the accumulated step point error can be represented by*

$$\begin{aligned} \alpha_{n+1,k} = {} & A_{qs}c(q)h^{q+2} + A_{q+1,s}c(q+1)h^{q+3} + \big(A_{q+2,s}c(q+2) + B_{qs}c(q)\big)h^{q+4} \\ & + \mathrm{O}(h^{q+5}), \end{aligned} \tag{3.27}$$

where for all $s \geqslant 1$ the row matrices A_{js} are defined by

$$\begin{aligned} A_{js} := {} & y^{(j+2)}(t_n)e_k^{\mathrm{T}} + y^{(j+2)}(t_{n-1})e_k^{\mathrm{T}}R + y^{(j+2)}(t_{n-2})e_k^{\mathrm{T}}R^2 + \cdots \\ & + y^{(j+2)}(t_{n-s+1})e_k^{\mathrm{T}}R^{s-1}, \quad j \geqslant q, \end{aligned} \tag{3.28a}$$

and where for $s = 1, 2, 3$ the row matrices B_{js} are defined by

$$\begin{aligned} B_{q1} &:= y^{(q+2)}(t_n)\frac{\partial f(y(t_n))}{\partial y}e_k^{\mathrm{T}}T, \\ B_{q2} &:= B_{q1} + y^{(q+2)}(t_{n-1})\frac{\partial f(y(t_n))}{\partial y}e_k^{\mathrm{T}}(RT + TR + S), \\ B_{q3} &:= B_{q2} + y^{(q+2)}(t_{n-2})\frac{\partial f(y(t_n))}{\partial y}e_k^{\mathrm{T}}(RTR + RS + SR + R^2T + TR^2). \end{aligned} \tag{3.28b}$$

Because the first $k-2$ columns of R^j vanish, (3.28a) shows that the last two entries of A_{js} do not vanish, so that the terms of order h^{q+2} and h^{q+3} in (3.27) are completely determined by the last two components of $\boldsymbol{c}(q)$ and $\boldsymbol{c}(q+1)$. Furthermore, because any two PSC methods with $b_{k-1} = \frac{1}{2}$ possess a matrix R with identical entries $R_{k,k-1}$ and $R_{k,k}$, we see that the last two entries of their A_{js} matrices also are identical. Hence, if their stage order q is equal, then the last two components of $\boldsymbol{c}(q)$ and $\boldsymbol{c}(q+1)$ may serve to compare their accuracy up to the order h^{q+3} terms.

The term of order h^{q+4} in (3.27) is much more complicated. Because it turns out that $A_{q+2,s}\boldsymbol{c}(q+2)$ can be neglected with respect to $B_{qs}\boldsymbol{c}(q)$, we concentrate on B_{qs}. Firstly, we observe that all k components of $\boldsymbol{c}(q)$ play a role in the size of $B_{qs}\boldsymbol{c}(q)$. Secondly, unlike the matrices A_{js}, the matrix B_{qs} may strongly differ for two different PSC methods, and thirdly, as is clear from (3.28b) its structure becomes increasingly complicated if s increases. Let us consider the matrix B_{q3}, that is, in the order h^{q+4} term of $\alpha_{n+1,k}$ we consider the accumulation of local errors over three steps:

$$B_{q3} = y^{(q+2)}(t_n)\frac{\partial f(y(t_n))}{\partial y}e_k^{\mathrm{T}}(T + RT + TR + S + RTR + RS + SR + R^2T + TR^2) + \mathrm{O}(h).$$

Then, we may define the following 'error constants' associated with the order h^{q+2}, h^{q+3} and h^{q+4} terms in the expansion of $\alpha_{n+1,k}$:

$$\begin{aligned} E(q) &= \|c_{k-1}(q), c_k(q)\|_\infty, \\ E(q+1) &= \|c_{k-1}(q+1), c_k(q+1)\|_\infty, \\ E(q+2) &= |e_k^{\mathrm{T}}(T + RT + S^* + R^2T + RS^* + S^*R)\boldsymbol{c}(q)|. \end{aligned}$$

For the PSC methods generated by the abscissa vectors $\boldsymbol{b}$ of Table 1, Table 2 lists the error constants $E(q)$, $E(q+1)$ and $E(q+2)$, where $q = k$ and $q = k+1$ for explicit and implicit PSC methods, respectively. As already observed, with each abscissa vector $\boldsymbol{b}$, we can associate both a predictor defined by $\{(2.9),\ T = O\}$ and a corrector defined by (3.5). Therefore, Table 2 presents both predictor and corrector values. Evidently, if $E(j) = 0$ for $j < p$, then the PSC method has step point order p. Hence, we shall call $E(p)$ the *principal error constant* (note that comparing the accuracy of two PSC methods by means of $E(j)$ values is only possible if their stage order q is identical).

Of course, the PSC method is only useful if the values of $|S_{ij}|$ and $|\delta_i|$ (in the case of implicit PSC methods) are sufficiently small and if the method is sufficiently stable. Therefore, Table 2 also lists an upper bound σ for $|S_{ij}|$ and the range of δ_i values. As to the stability, it turns out that in many cases the stability boundary β defined in Section 2.6 is zero. However, if we relax the definition of the stability region by allowing that the eigenvalues of $M(z)$ are bounded by $1+\varepsilon$ with $0 < \varepsilon \ll 1$, then we obtain quite substantial stability boundaries β^*, even for extremely small ε. For $\varepsilon = 10^{-6}$ we found the values

Table 2
Characteristics of the PC pairs generated by the abscissa vectors $\boldsymbol{b}$ from Table 1

		Predictor						Corrector						
k	$\boldsymbol{b}$	$E(k)$	$E(k+1)$	$E(k+2)$	p	σ	β^*	$E(k+1)$	$E(k+2)$	$E(k+3)$	p	σ	δ_i	β^*
4	(3.15)	0	3.5E−5	4.9E−3	5	3.3	0.79	3.5E−5	3.1E−5	4.4E−4	5	3.5	(−0.000, 0.036)	0.86
4	(3.17)	6.2E−4	3.7E−4	1.2E−2	4	21	0.37	0	1.4E−6	6.6E−4	6	4.3	(0.014, 0.146)	0.47
5	(3.16)	0	1.9E−6	9.3E−4	6	4.0	0.85	1.9E−6	9.3E−7	4.2E−5	6	1.5	(−0.000, 0.058)	1.08
5	(3.18)	2.9E−4	9.8E−5	6.5E−3	5	63	0.90	0	4.7E−7	3.0E−4	7	9.3	(−0.018, 0.097)	0.59
6	(3.19)	0	0	1.3E−4	8	30	0.74	0	1.2E−10	5.2E−6	8	7.1	(−0.008, 0.041)	1.01
6	(3.22)	8.9E−8	7.0E−8	1.3E−4	6	27	0.74	0	0	4.9E−6	9	6.6	(−0.006, 0.041)	1.01
7	(3.20)	0	0	4.4E−5	9	65	0.80	0	7.3E−11	1.5E−6	9	13	(−0.007, 0.036)	0.98
7	(3.23)	8.8E−8	5.4E−8	5.1E−4	7	67	0.80	0	0	1.6E−6	10	15	(−0.002, 0.035)	1.01
8	(3.21)	0	0	3.9E−8	10	319	0.78	0	2.5E−12	1.4E−8	10	49	(−0.022, 0.040)	0.66
8	(3.24)	2.2E−9	1.6E−9	5.2E−7	8	260	0.78	0	0	5.5E−10	11	42	(−0.005, 0.044)	0.65

as listed in Table 2. In actual computation these values turn out to be sufficiently large in the sense that the stepsize is prescribed by accuracy and not by stability.

4. Implementation aspects

When implementing the PSC methods constructed above, we have to decide about the computation of the starting vector $\boldsymbol{Y}_0$ needed to start the recursion (2.1), the local error estimate, and the stepsize strategy. In the case of implicit PSC methods, we always started the iteration by the predictor formula associated with the abscissa vector of the implicit PSC method.

The starting vector $\boldsymbol{Y}_0 \approx (\boldsymbol{y}(t_0 + b_i h))$ is most conveniently computed by means of a one-step method. In all our experiments, we computed the stage values of $\boldsymbol{Y}_0$ by means of one (accepted) step of the 7th-order Runge–Kutta–Nyström method of Dormand and Prince [3]. Note that these k stage values can be computed in parallel, so that effectively only one Dormand and Prince step is needed. Furthermore, we remark that Runge–Kutta–Nyström methods can be applied for negative stepsizes, so that negative values of b_i are allowed.

The numerical experiments presented in Section 5.2 use a variable stepsize implementation of PSC methods, so that we shall briefly discuss the stepsize procedure applied in this paper. The abscissae obtained in the preceding sections assume constant stepsizes, so that we should either allow h to be variable in (2.1) and determine abscissae corresponding to nonconstant stepsizes h, or we should replace in (2.1) the stage vector $\boldsymbol{Y}_n$ corresponding to the points $t_{n-1} + a_i h$ by a new stage vector $\boldsymbol{V}_n$ corresponding to the points $t_{n-1} + a_i h_{\text{new}}$. The first option is not feasible, because it would mean that the process described above should be performed each time the stepsize changes. The second option, however, is quite straightforward. Let

$$V_n = (P \otimes I)Y_n + h^2(Q \otimes I)F(Y_n), \qquad \theta = \frac{h_{\text{new}}}{h}, \tag{4.1}$$

where the k-by-k matrices P and Q are such that V_n represents a numerical approximation to the exact solution values $y(t_{n-1} + a_i h_{\text{new}}) = y(t_n + b_i h_{\text{new}})$. Proceeding as in Section 2.2, we find the conditions

$$Pb^j + j(j-1)Qb^{j-2} = (\theta b)^j, \quad j = 0, \ldots, q \tag{4.2}$$

(cf. (2.6)). If $q \geqslant k+1$, then the interpolation error is of order $k+2$ in h. In order to keep the entries of P and Q of acceptable magnitude for larger values of k, we should not allow P and Q to be full matrices. The minimal number of nonzero columns needed in the matrix (P, Q) to achieve order $k+2$ interpolation is $k+2$. Since the last two stage values in Y_n are of increased accuracy and because of the factor h^2 in front of $F(Y_n)$, it follows from (4.1) that it is natural to set the entries in the first $k-2$ columns of P equal to zero. Thus, the interpolation formula (4.1) is based on the last two y-values and on all f-values available from the preceding step. Let P^* be the k-by-2 matrix containing the last two columns of P. Then, (4.2) is equivalent with the condition

$$(P^*, Q)U = W, \quad U := \begin{pmatrix} 1 & b_{k-1} & b_{k-1}^2 & b_{k-1}^3 & \cdots & b_{k-1}^{k+1} \\ 1 & b_k & b_k^2 & b_k^3 & \cdots & b_k^{k+1} \\ \mathbf{0} & \mathbf{0} & 2e & 6b & \cdots & k(k+1)b^{k-1} \end{pmatrix},$$
$$W := \left(e, \theta b, \ldots, (\theta b)^{k+1}\right). \tag{4.3}$$

Hence, the matrices P and Q follow from the formula $(P^*, Q) = WU^{-1}$. The magnitude of the entries of P and Q increases strongly with θ and k. However, for $\theta = \frac{3}{2}$ and $k = 8$ (with abscissae as defined in Table 1), their magnitude is still acceptable (six in the range 12–60, the remaining less than 4). We remark that if the interpolation formula (4.1) is based on the last two f-values and on all y-values, then the magnitude of the entries in P and Q is unacceptably large (up to about 54000 for $\theta = \frac{3}{2}$ and $k = 8$).

As soon as we change the stepsize, we apply the interpolation procedure described above. In this way, we achieve that we may always use the constant stepsize formula (2.1). However, it also means that each stepsize change implies the evaluation of $F(V_n)$. We can reduce these extra costs by applying a stepsize strategy which keeps the number of stepsize changes low.

Apart from an interpolation procedure, we also need an error estimator in the case of stepsize changes. Observing that the solution values $y_{n,k-1}$, $y_{n+1,k}$ and $y_{n+1,k-1}$ correspond with t-values at distance $\frac{1}{2}h$, we may use the Numerov formula

$$z_{n+1} = \tfrac{1}{2}\left(y_{n,k-1} + y_{n+1,k-1} - \tfrac{1}{48}h^2\left(f(y_{n,k-1}) + 10f(y_{n+1,k}) + f(y_{n+1,k-1})\right)\right) \tag{4.4}$$

as a fourth-order reference solution for appreciating the quality of the step point value $y_{n+1,k}$. Note that this formula only uses already computed values. In choosing the new stepsize, we adopted the standard procedure used in ODE solvers (see, e.g., [7, p. 167]) with a slight modification in order to keep stepsize changes to a minimum. Let tol be a given tolerance parameter and define

$$\text{err} := \left\| \frac{z_{n+1} - y_{n+1,k}}{\max\{|y_{n+1,k}|, 10^{-6}\}} \right\|_\infty, \qquad h^* = h \cdot \min\left\{1.5, \max\left\{0.5, 0.8\sqrt[5]{\frac{\text{tol}}{\text{err}}}\right\}\right\}.$$

This leads to the following stepsize strategy:

if $0.01\,\mathrm{tol} < \mathrm{err} < \mathrm{tol}$ **then** $h_{\mathrm{new}} = h$: perform next step,

if $\mathrm{err} \leqslant 0.01\,\mathrm{tol}$ **then** $h_{\mathrm{new}} = h^*$: perform next step,

if $\mathrm{err} \geqslant \mathrm{tol}$ **then** $h_{\mathrm{new}} = h^*$: redo step.

5. Numerical experiments

In this section we illustrate the performance of the PSC methods generated by the abscissae of Table 1. From now on, a PSC method is understood to be determined by (i) an abscissa vector $\boldsymbol{b}$ defining the predictor–corrector pair, and (ii) a PE(CE)^m or PE(CE)^{m}C iteration strategy. We performed many experiments on well-known test problems taken from the literature of which a few typical performance tests will be reproduced in the tables of results (Tables 3–9). In these tables, the accuracy is defined by the number of correct digits Δ at the end point (that is, the maximal absolute end point error is written as $10^{-\Delta}$, and the total number of steps and the total number of sequential right-hand sides needed in the integration process is denoted by N and M, respectively. Furthermore, we present the effective order of accuracy p^*, based on the two last computed results in the case of constant stepsize experiments and on the minimal and maximal error tolerance results in the case of variable stepsize experiments. The corresponding formulas for p^* are respectively given by

$$p^* := \frac{\Delta(2h_{\min}) - \Delta(h_{\min})}{\log_{10}(2)} \quad \text{and} \quad p^* := \frac{\Delta(\mathrm{tol}_{\min}) - \Delta(\mathrm{tol}_{\max})}{M(\mathrm{tol}_{\min}) - M(\mathrm{tol}_{\max})}. \tag{5.1}$$

5.1. *Selection of the most efficient method*

First we want to know which PSC method is the most efficient one. It may be expected that for a given number of iterations m, the PE(CE)^{m}C mode yields the same or a higher accuracy than the PE(CE)^m mode (note that these modes are equally expensive). This claim was carefully checked and turned out to be true for all problems we tested. Furthermore, we observed that the accuracy did not improve anymore by performing more than two iterations. Therefore, we only give results for the PEC and the P(EC)^2 mode. In order to see clearly the algorithmic properties of the methods, we applied them with fixed stepsizes. A comparison is presented in the Tables 3, 4 and 5, respectively for methods with 4, 6 and 7 computational stages, that is, methods requiring 4, 6 and 7 processors. For this comparison, we chose the TWOB problem [7, p. 236] on the interval $[0, 20]$ with eccentricity $\varepsilon = 0.5$, because the performance of the various methods on this problem turned out to be representative for a large class of problems.

5.1.1. *Four-processor methods*

From Table 1 we selected the four abscissa vectors (3.15)–(3.18) which generate PSC methods with 4 computational stages. The error constants listed in Table 2, indicate that (3.16) and (3.18) generate the most accurate predictor and corrector, respectively. Therefore, we expect that these abscissa vectors generate the most efficient PEC and P(EC)^2 methods. This conclusion is confirmed by the results of Table 3 (and by many other examples we tested). Taking into account that P(EC)^2 is about twice as costly as PEC, this table clearly shows that {(3.16), PEC}, {(3.16), P(EC)^2}, and {(3.18), P(EC)^2} are

Table 3
(Δ, N)-values for PSC methods with 4 computational stages applied to TWOB

b	Mode	p	80	160	320	640	1280	2560	5120	p^*
(3.15)	PEC	5	0.4	2.0	4.4	6.2	7.8	9.6	11.8	7.0
(3.16)		6	**0.8**	**3.2**	**4.5**	6.5	8.6	10.7	12.8	7.0
(3.17)		4	0.1	1.4	3.5	5.9	7.3	8.9	10.6	5.8
(3.18)		5	0.3	2.1	3.6	5.6	7.7	9.8	11.9	7.0
(3.15)	P(EC)2	5	1.4	3.1	4.9	6.6	8.1	9.6	11.1	5.0
(3.16)		6	1.4	4.2	**7.1**	**8.9**	**12.2**	**12.8**	14.5	5.7
(3.17)		6	0.9	2.6	4.4	6.4	8.5	10.6	12.7	7.0
(3.18)		7	1.3	3.0	6.1	7.9	10.1	12.4	**14.8**	8.0

Table 4
(Δ, N)-values for PSC methods with 6 computational stages applied to TWOB

b	Mode	p	80	160	320	640	1280	2560	5120	p^*
{(3.19), (3.22)}	PEC	{8, 6}	**0.8**	**4.0**	**6.4**	8.4	10.6	12.9	15.3	7.9
{(3.20), (3.23)}		{9, 7}	0.6	3.8	6.2	**8.8**	**11.5**	**14.2**	**17.0**	9.1
{(3.19), (3.22)}	P(EC)^2	{8, 9}	2.9	5.1	7.1	9.8	12.5	15.2	17.8	8.7
{(3.20), (3.23)}		{9, 10}	1.7	5.1	8.0	10.7	13.6	16.6		10.0

the most efficient methods respectively in the low accuracy range (1–5 digits, say), in the middle-high accuracy range (5–15 digits), and in the extremely high accuracy range (15 or more digits).

5.1.2. *Six-processor methods*

PSC methods with 6 computational stages are generated by (3.19), (3.20), (3.22) and (3.23). Table 2 indicates that (3.20) furnishes the most accurate predictor and (3.23) the most accurate corrector. Hence, we anticipate that these abscissa vectors provide the most efficient PEC and P(EC)^2 methods. However, it turns out that the methods generated by (3.19) and (3.22) produce the same results as the methods generated by (3.22) and (3.23). This can be explained by observing that the principal error constants of (3.19) and (3.22) are extremely small (see Table 2), so that the characteristics of the methods generated by (3.19) and (3.20) closely resemble the characteristics of the methods generated by (3.22) and (3.23), respectively. The results presented in Table 4 are typical for a large number of experiments that we have carried out. From these figures we draw the conclusion that {(3.19), PEC} or {(3.22), PEC} is most efficient in the low accuracy range, {(3.20), PEC} or {(3.23), PEC} is most efficient in the middle-high accuracy range, and {(3.20), P(EC)^2} or {(3.23), P(EC)^2} is most efficient in the extremely high accuracy range.

Table 5
(Δ, N)-values for PSC methods with 7 computational stages applied to TWOB

b	Mode	p	80	160	320	640	1280	2560	p^*
(3.21)	PEC	10	**1.5**	5.0	**8.2**	**11.6**	**15.4**		11.0
(3.24)		8	0.8	**5.2**	7.7	9.9	12.7	15.7	10.0
(3.21)	P(EC)^2	10	3.3	6.0	9.6	12.9	16.7		13.0
(3.24)		11	1.9	5.0	8.3	11.6	15.0		11.3

Table 6
Most efficient PSC methods

Processors	$0 \leqslant \Delta \leqslant 5$	$5 \leqslant \Delta \leqslant 15$	$\Delta \geqslant 15$
4	{(3.16), PEC}	{(3.16), P(EC)^2}	{(3.18), P(EC)^2}
6	{(3.19), PEC}	{(3.20), PEC}	{(3.23), P(EC)^2}
7	{(3.21), PEC}	{(3.21), PEC}	{(3.21), PEC}

5.1.3. Seven-processor methods

All error constants listed in Table 2 for the 7-computational-stage methods associated with (3.21) and (3.24) are extremely small. Therefore, it is dangerous to base conclusions on their magnitude, because the higher-order error constants may play a more dominant role unless we use unrealistic small stepsizes. On the basis of many numerical experiments, we found that in general (3.21) generates more efficient PSC methods than (3.24). A typical performance is listed in Table 5.

5.1.4. Summary of recommended methods

For a few accuracy ranges $[\Delta_1, \Delta_2]$, Table 6 summarizes the most efficient PSC methods of this paper for 4, 6 and 7 processor computer systems.

5.2. Comparison with other codes

In this section, we compare the variable step version of our most powerful PSC method, that is, the 10th-order method {(3.21), PEC} using automatic stepsize control as described in Section 4, with one of the best sequential code available in the literature, viz. the 7th-order variable step code DOPRIN [7], and with another parallel code, viz. the 12th-order variable step PIRKN code of Sommeijer [10]. We present the total number of sequential right-hand sides M needed to produce a given number of correct digits Δ (including the right-hand sides to generate the starting values by means of DOPRIN). The M values were obtained by running the DOPRIN. PIRKN and PSC codes with tolerances 10^{-1}, $10^{-2}, \ldots$ and by linear interpolation of the Δ and $\log_{10}(M)$ values produced. Tables 7–9 show results for the two-body orbit problem from the Toronto test set [8] (see also [10, (3.1)] on the interval [0, 20] with eccentricity $\varepsilon = 0.9$, the often used Fehlberg stability test problem (see [10, (3.2)]) on the interval $[\sqrt{\frac{1}{2}\pi}, 10]$, and the PLEI problem [7, p. 237]. For these problems, the initial step used by DOPRIN and by {(3.21), PEC} was $h_0 = 0.01$, 0.1 and 0.01, respectively.

Table 7
(Δ, M)-values for the orbit problem

Δ	5	6	7	8	9	10	11	12	13	14	15	p^*
DOPRIN	1365	1598	1782	2515	3267	3938	4587	5598	6754	8049	13277	8.7
PIRKN	475	521	572	654	806	994	1205	1445	1698	1986	2239	15.9
{(3.21), PEC}	294	335	401	483	585	720	896	1122	1401	1751	2189	12.4

Table 8
(Δ, M)-values for the Fehlberg problem

Δ	5	6	7	8	9	10	11	12	13	14	15	16	17	p^*
DOPRIN	886	1276	1793	2545	3485	4759	6423	8779	12217	16746	23121	32162	44571	7.2
PIRKN	362	427	508	604	724	868	1041	1249	1499	1799	2185	2653	3216	12.6
{(3.21), PEC}	154	193	238	277	330	409	505	613	740	889	1069	1270	1508	12.1

Table 9
(Δ, M)-values for the PLEI problem

Δ	5	6	7	8	9	10	11	12	13	14	15	p^*
DOPRIN	1444	1743	2154	2674	3165	3673	4380	5668	7691	10853	15295	9.8
PIRKN	511	586	680	795	938	1116	1337	1581	1864	2199	2561	14.3
{(3.21), PEC}	312	368	436	540	666	807	991	1229	1446	1987	2519	11.0

6. Concluding remarks

In this paper, we constructed parallel Störmer–Cowell type methods (PSC methods) with orders ranging from $p = 4$ until $p = 11$ for parallel computer systems with 4 until 7 processors. Of these PSC methods the 10th-order, 7-processor method turns out to be most effective for high-precision orbit computations. This method was compared with DOPRIN, one of the most efficient sequential code currently available, and with PIRKN, a 12th-order, 6-processor code. In terms of the total number of right-hand side evaluations needed in the integration process, the speed-up of the PSC method with respect to DOPRIN ranges from 4 in the low accuracy range up to 30 in the high accuracy range, and with respect to PIRKN, the PSC method is at least equally efficient and at best twice as fast.

References

[1] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover, 1964.
[2] J.C. Butcher, The Numerical Analysis of Ordinary Differential Equations, Runge–Kutta and General Linear Methods, Wiley, New York, 1987.
[3] J.R. Dormand, P.J. Prince, New Runge–Kutta algorithms for numerical simulation in dynamical astronomy, Celestial Mech. 18 (1978) 223–232.

[4] S. Fehlberg, S. Filippi, J. Gräf, A Runge–Kutta–Nyström formula pair of order 10(11) for differential equations of the form $y'' = f(x, y)$, Z. Angew. Math. Mech. 66 (1986) 265–270.

[5] S. Filippi, J. Gräf, A Runge–Kutta–Nyström formula pair of order 11(12) for differential equations of the form $y'' = f(x, y)$, Computing 34 (1985) 271–282 (in German).

[6] S. Filippi, 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 $y'' = f(x, y)$, J. Comput. Appl. Math. 14 (1986) 361–370.

[7] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations, Vol. I. Nonstiff Problems, Springer, Berlin, 1987.

[8] T.E. Hull, W.H. Enright, B.M. Fellen, A.E. Sedgwick, Comparing numerical methods for ordinary differential equations, SIAM J. Numer. Anal. 9 (1972) 603–637.

[9] W.M. Lioen, J.J.B. de Swart, W.A. van der Veen, Test set for IVP solvers, CWI Report NM-R9615, Amsterdam. Available at http://www.cwi.nl/cwi/projects/IVPtestset/.

[10] B.P. Sommeijer, Explicit high-order Runge–Kutta–Nyström methods for parallel computers, Appl. Numer. Math. 13 (1993) 221–240.

[11] P.J. van der Houwen, E. Messina, Parallel Adams methods, to appear in J. Comput. Appl. Math.