A new class of highly accurate solvers for ordinary differential equations

Andreas Glaser and Vladimir Rokhlin
Technical Report YALEU/DCS/TR-1382
July 17, 2007

We introduce a new class of numerical schemes for the solution of the Cauchy problem for non-stiff ordinary differential equations (ODEs). Our algorithms are of the predictor-corrector type; they are obtained via the decomposition of the solutions of the ODEs into combinations of appropriately chosen exponentials, whereas the classical schemes are based on the approximation of solutions by polynomials. The resulting schemes have the advantage of significantly faster convergence, given fixed lengths of predictor and corrector vectors. The performance of the approach is illustrated via a number of numerical examples.

A new class of highly accurate solvers for ordinary differential equations

Andreas Glaser and Vladimir Rokhlin
Technical Report YALEU/DCS/TR-1382
July 17, 2007

The authors were supported in part by the U.S. Department of Defense under ONR Grant #N00014-07-1-0711 and AFOSR Grants #FA9550-06-1-0197 and #FA9550-06-1-0239.
Approved for public release: Distribution is unlimited.
Keywords: *Ordinary Differential Equations, Predictor-Corrector*

1 Introduction

Numerical algorithms for the solution of ordinary differential equations (ODEs) are an essential part of many scientific computations. As a consequence, a broad variety of such algorithms exists and their construction and analysis are a well-established fields (see, for example, [10, 11, 14, 2, 15]).

The existing algorithms can be grouped into direct solvers such as Runge-Kutta and multi-step methods [10] and schemes which improve the solution iteratively as, for example, Richardson extrapolation [15] and deferred correction [6, 13, 12]. Typically, the construction of direct solvers is based on the assumption that the solution of the ODE can be locally approximated by a polynomial, with the degree of the polynomial determining the order of accuracy of the scheme (see, for example, [15]). Explicit variants of such schemes are applicable to non-stiff problems; the ones in wide use appear to be of orders 2 to 12. Iterative methods (such as deferred corrections) have been published that are of essentially arbitrary order.

While the existing direct and iterative techniques provide a reliable tool for many applications, they tend to be very expensive for solving ODEs with high accuracy. Direct methods require very small step-sizes to achieve high accuracy, while iterative methods involve repeated solution of the ODE (or its slight modification).

In the early stages of numerical computing there have been attempts to construct direct schemes based on the assumption that the solution which is to be computed can be represented as linear combination of exponentials or trigonometric functions [7, 3], an approach similar to the one taken in this paper. However, these endeavors never resulted in algorithms competitive in practical applications.

In this paper we introduce a new class of algorithms for solving the Cauchy problem for non-stiff ordinary differential equations (ODEs) which have a significantly faster rate of convergence than the classical schemes. The schemes are of the predictor-corrector type and are based on the approximation of the solution by a linear combination of a finite collection of exponentials, chosen via an appropriate "skeletonization" procedure (see [9, 4]). The latter representation is combined with standard numerical tools (such as least squares approximation) to construct efficient extrapolation and integration formulae. The starting values required to initialize the predictor-corrector scheme are computed via a spectral deferred correction method (see [6]).

It should be observed that the algorithms presented in this paper are not convergent in the classical sense; each particular scheme quickly converges to a certain predetermined precision, after which its accuracy stops improving as a function of the number of nodes in the discretization. While this can be viewed as a drawback from the theoretical point of view, in practical calculations the precision is always fixed (in double precision arithmetic, it is about 16 digits). Thus, the performance of a scheme designed with 16 digits is indistinguishable from that of a classically convergent scheme, as long as the calculations are performed in double precision arithmetic.

Throughout this paper, it is assumed that the Cauchy problem to be solved is in the form

$$\begin{aligned} \frac{d\varphi}{dt}(t) &= F(t,\varphi(t)), &(1)\\ \varphi(a) &= \varphi_0, &(2) \end{aligned}$$

where $t \in [a,b]$, $\varphi_0 \in \mathbb{C}^m$, $\varphi : \mathbb{R} \to \mathbb{C}^m$ and $F : \mathbb{R} \times \mathbb{C}^m \to \mathbb{C}^m$ is assumed to be sufficiently smooth. As is well known, any initial value problem involving ODEs with higher derivatives can be reduced to this standard form. For the convenience of presentation we assume that $m = 1$, as the generalization of the construction for the case of arbitrary m is straightforward.

The structure of this paper is as follows. In Section 2, we summarize several well-known mathematical and numerical tools to be used in this paper, while in Section 3, we discuss the specific analytical and mathematical apparatus required for the construction of the numerical solvers. Section 4 describes the numerical solvers, and in Section 5 we report a number of numerical experiments, including the establishment of stability and accuracy properties.

2 Mathematical and numerical preliminaries

In this section we summarize several well-known facts to be used in the remainder of this paper. All of these can be found, for example, in [1, 10, 11, 5, 4]. Throughout this paper, the set $\{t_1, \dots, t_n\}$ with $t_1 = a$, $t_2 = b$ and $t_i = a + (i-1)h$ will be called a equidistant discretization of the interval [a,b] with step-size h. Furthermore, the derivative of a function $f : \mathbb{R} \to \mathbb{R}$ will be denoted by f', the conjugate transpose is denoted by *, the complex modulus or absolute value is denoted by $|\cdot|$, and $||\cdot||_2$ denotes the Euclidian norm of a vector or the corresponding operator norm of a matrix.

2.1 Linear predictor-corrector methods

A predictor-corrector method solves the initial value problem (1) by advancing the solution step by step via extrapolation (predictor step) and integration (corrector step), based on the k previous time steps. Given the values of φ and φ' at the k equidistant nodes $t_1, t_2, \dots, t_k$ the predictor step approximates the value of φ at the next time step t_{k+1} by the linear extrapolation formula

$$\varphi(t_{k+1}) = \sum_{i=1}^{k} \left[p_i \varphi(t_i) + p_{k+i} F(t_i, \varphi(t_i)) \right], \tag{3}$$

where $p_1, \dots, p_{2k} \in \mathbb{R}$. The result is used to compute the value of $\varphi'(t_{k+1})$ via equation (1). The corrector step then recomputes $\varphi(t_{k+1})$ by the integration formula

$$\varphi(t_{k+1}) = \sum_{i=1}^{k} \left[c_i \varphi(t_i) + c_{k+i} F(t_i, \varphi(t_i)) \right] + c_{2k+1} F(t_{k+1}, \varphi(t_{k+1})), \tag{4}$$

where $c_1, \ldots, c_{2k+1} \in \mathbb{R}$, and the value of $\varphi'(t_{k+1})$ is updated via equation (1). One predictor step can be followed by several corrector steps.

In accordance with conventions in the literature, we refer to a numerical scheme which solves equation (1) by computing $\varphi(t_k + 1)$ from $\varphi(t_1), \ldots, \varphi(t_k)$ via one predictor step (3) and m corrector steps (4) as k-step $PE(CE)^m$ method (see [15]). For their initialization k-step $PE(CE)^m$ methods require k starting values.

Remark 1 Most predictor-corrector methods in use ignore the values of φ in formulae (3), (4) and use only the information about the derivative $F(t, \varphi)$. In this paper, however, we will use formulae (3), (4) in their full generality.

Remark 2 The most popular predictor-corrector methods appear to be Adams-Bashforth-Moulton schemes (see, for example, [10, 15]). The predictor and corrector formulae in these schemes are based on approximating φ' by a polynomial.

2.2 Accuracy and stability of numerical ODE solvers

The two characteristics which are generally used to describe the performance of a numerical ODE solver are its order of accuracy and its stability region (see, for example, [10, 11, 14]). Let $\widetilde{\varphi}$ denote the solution to the initial value problem (1), (2), obtained via a numerical solver at the equidistant discretization $t_1, \ldots, t_n$ of $[a, b]$ with step-size h. The underlying method is said to be of order of accuracy p, if for any sufficiently smooth F in equation (1) there exists a constant $M > 0$ such that

$$|\varphi(b) - \widetilde{\varphi}(b)| < M\, h^p, \tag{5}$$

for any choice of h, that is sufficiently small.

The stability of a numerical ODE solver is generally determined by analyzing its behavior on test problems of the form

$$\begin{aligned} \varphi'(t) &= \lambda \varphi(t), & (6) \\ \varphi(0) &= 1 & (7) \end{aligned}$$

where $\lambda \in \mathbb{C}$. A numerical scheme, which computes the solution $\widetilde{\varphi}$ to this problem at the equidistant discretization $t_1, \ldots, t_n$ with *step-size one*, is said to be stable for λ, if

$$\widetilde{\varphi}(t_i) \leq 1, \tag{8}$$

for $i = 1, \ldots, n$. The set of values of λ for which a numerical method is stable is called its *stability domain.*

The stability of a k-step $PE(CE)^m$ method, in particular, can be determined by computing the eigenvalues of the matrix B which corresponds to the the linear mapping $B : \mathbb{C}^k \to \mathbb{C}^k$, defined by

$$\begin{pmatrix} \widetilde{\varphi}(t_1) \\ \widetilde{\varphi}(t_2) \\ \vdots \\ \widetilde{\varphi}(t_k) \end{pmatrix} \mapsto \begin{pmatrix} \widetilde{\varphi}(t_2) \\ \widetilde{\varphi}(t_3) \\ \vdots \\ \widetilde{\varphi}(t_{k+1}) \end{pmatrix}, \tag{9}$$

where $\widetilde{\varphi}(t_1), \ldots, \widetilde{\varphi}(t_k)$ are arbitrary starting values and $\widetilde{\varphi}(t_{k+1})$ is the result of using the predictor-corrector method to take a step of length one for the solution of equation (6). If the biggest eigenvalue of B is less than one then the method is stable for λ. The matrix B is of the form

$$B = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ b_1 & b_2 & \cdots & b_k \end{pmatrix}, \tag{10}$$

where the coefficients $b_1, \ldots, b_k \in \mathbb{C}$ depend on the given predictor-corrector method and on λ. Specifically, $b_i = \widetilde{\varphi}_i(t_{k+1})$, where $\widetilde{\varphi}_i(t_{k+1})$ is the result of applying the given predictor-corrector method to the starting values $\widetilde{\varphi}_i(t_1), \ldots, \widetilde{\varphi}_i(t_k)$, chosen as

$$\widetilde{\varphi}_i(t_j) = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{if } j \neq i \end{cases}. \tag{11}$$

Remark 3 The matrix B is the transpose of a companion matrix and has the characteristic polynomial (see, for example, [8])

$$-b_1 + -b_2 z + -b_3 z^2 + \cdots - b_k z^{k-1} + z^k = 0. \tag{12}$$

In the literature (see, for example, [11, 14]) the stability of a multi-step method is often determined by checking if the largest root of equation (12) is less or equal to one. Obviously, this is equivalent to the stability criterion given above.

Finally, suppose as before, that $\widetilde{\varphi}$ denotes the solution to (6), (7), obtained via a numerical method at the equidistant discretization $t_1, \ldots, t_n$ of $[a, b]$ with step-size one. For a specified precision ε we define the method's accuracy domain to be the set of all λ, for which

$$\sqrt{\frac{\sum_{i=1}^n |\varphi(t_i) - \widetilde{\varphi}(t_i)|^2}{\sum_{i=1}^n |\varphi(t_i)|^2}} < \varepsilon, \tag{13}$$

where $\varphi : \mathbb{C} \mapsto \mathbb{C}$, $z \mapsto e^{\lambda z}$ denotes the analytical solution to equations (6), (7).

Remark 4 Obviously, increasing the number of discretization steps of a given interval will increase the frequency domain to which a numerical method applies. In general, changing the step-size of a numerical scheme for the solution of the problem (6), (7) from one to h is equivalent to changing the exponent λ in equation (6) to λh, as the dilation $\tau = t/h$ transforms equation (6) into

$$\frac{\partial}{\partial \tau} \varphi(\tau) = \lambda h \, \varphi(\tau). \tag{14}$$

In the case of the variable step-size h, stability and accuracy regions therefore apply to the product λh.

Remark 5 A standard way of describing the performance of a numerical scheme for a given accuracy is in terms of the *number of steps per wavelength.* If the accuracy domain of the scheme for the precision ε contains the semi-disk

$$S = \left\{ \lambda \in \mathbb{C} \mid \operatorname{Re}(\lambda) \leq 0, \ |\lambda| \leq |\widetilde{\lambda}| \right\}, \tag{15}$$

then the scheme is said to require

$$\frac{2\pi}{|\widetilde{\lambda}|} \tag{16}$$

steps per wavelength to achieve the accuracy ε.

2.3 SVD and least squares

Given the linear system

$$Ax = b, \tag{17}$$

where $A \in \mathbb{C}^{m \times n}$ and $b \in \mathbb{C}^m$, any vector $x \in \mathbb{C}^n$ which minimizes

$$||Ax - b||_2 \tag{18}$$

is called a least squares solution of system (17). If X is the set of least squares solutions of (17) then X contains one unique element of minimum L_2-norm

$$x_{LM} = \operatorname{argmin}\{||x||_2 \mid x \in X\}, \tag{19}$$

which is called the minimum norm least squares solution. The singular value decomposition (SVD) provides an explicit expression for x_{LM}. If $A = U\Sigma V^*$ is the SVD of A, i.e. $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are orthonormal and $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal, then

$$x_{LM} = \sum_{i=1}^{r} \frac{u_i^* b}{\sigma_i} v_i, \tag{20}$$

where $(u_1, \ldots, u_m)$ are the columns of U, $(v_1, \ldots, v_n)$ are the columns of V, $(\sigma_1, \ldots, \sigma_r)$ are the positive diagonal elements of Σ, and r is the rank of A.

In numerical applications it is reasonable to define the rank of matrix A to precision ε. Given $\varepsilon > 0$ and the singular values $(\sigma_1, \ldots, \sigma_{min(m,n)})$ of A, sorted in decreasing order, the numerical rank of precision ε of A is defined as $r \in \mathbb{N}$, such that $\sigma_r \geq \varepsilon$ and $\sigma_{r+1} < \varepsilon$. Accordingly, we define the minimum norm least squares solution of precision ε of system (17) as

$$x_{LM}(\varepsilon) = \sum_{i=1}^{r(\varepsilon)} \frac{u_i^* b}{\sigma_i} v_i, \tag{21}$$

where $r(\varepsilon)$ is the numerical rank of precision ε of A and all other quantities are as in equation (20).

Given an algorithm for the computation of the SVD, the minimum norm least squares solution $x_{LM}(\varepsilon)$ can be computed via (21). However, besides this obvious SVD-based algorithm, there exist various other schemes for the computation of $x_{LM}(\varepsilon)$, which are usually faster; an example is the complete orthogonal factorization described in Chapter 5 of [8].

2.4 Matrix interpolation (Skeletonization)

The following lemma will be used for the approximation of exponentials of bounded frequency. It is often referred to as *skeletonization* and is given by [9, 4] in slightly more general form.

Lemma 6 (Skeletonization) *Given a matrix $A \in \mathbb{C}^{m\times n}$ with columns $(a_1, \ldots, a_n)$, and $k \in \mathbb{N}$ such that $1 \le k < min(m,n)$, there exists a selection of k columns of A such that*

$$A = (a_{n_1}, \ldots, a_{n_k})T + X, \tag{22}$$

where all elements of $T \in \mathbb{C}^{k\times m}$ have magnitude less than one and the operator norm of $X \in \mathbb{C}^{m\times n}$ is bounded by the $(k+1)$th singular value $\sigma_{k+1}(A)$ of A as follows

$$||X||_2 \le \sigma_{k+1}(A)\sqrt{1 + k(min(m,n) - k)}. \tag{23}$$

In other words, the lemma states that A can be approximated by interpolating between k of A's columns, if the $(k+1)$th singular value of A is small, i. e. if k is close to the numerical rank of A. The fact that all elements in the interpolation matrix T have magnitude less than one guarantees that this interpolation is stable.

Remark 7 The factorization (22) can typically be computed in $O(nmk)$ operations (the worst case is $O(mnl)$ operations, where $l = \min(m,n)$) by the algorithms described in [9, 4].

2.5 Spectral deferred correction methods

Spectral deferred correction methods solve the initial value problem (1), (2) by starting with a low accuracy approximation to the solution and improving it iteratively (see [6, 13, 12]). During each iteration the error ε of the current approximation φ^j is computed via the formula

$$\varepsilon(t) = \varphi(a) + \int_a^t F(\tau, \varphi^j(\tau))d\tau - \varphi^j(t), \tag{24}$$

which is based on the integral form (Picard equation) of equations (1), (2),

$$\varphi(t) = \varphi(a) + \int_a^t F(\tau, \varphi(\tau))d\tau. \tag{25}$$

The approximation $\varphi^j(t)$ is then updated via the formula

$$\varphi^{j+1}(t) = \varphi^j(t) + \gamma(t), \tag{26}$$

where $\gamma(t)$ has to satisfy the integral equation

$$\gamma(t) = \int_a^t [F(\tau, \varphi^j(\tau) + \gamma(\tau)) - F(\tau, \varphi^j(\tau))]d\tau + \varepsilon(t), \tag{27}$$

which results from combining (25) with (24) and assuming that φ_{j+1} satisfies (25).

Spectral deferred correction algorithms solve equation (27) by standard ODE solvers such as Euler's method, and evaluate the integral in (24) by spectral integration (see [6]). A typical choice for the discretization of the integration interval are Gaussian or Tchebycheff nodes (see [6, 13]). A discussion of the stability and convergence properties of these schemes can be found in [6, 12]. In Section 3.4 we construct a spectral deferred correction scheme for equidistant nodes.

2.6 A second-order Runge-Kutta method

The second-order Runge-Kutta method solves the initial-value problem (1), (2) on the interval $[t_0, t_0 + L]$ by taking a sequence of n steps:

$$\begin{aligned} t_{i+1} &= t_i + h, \\ k_{i+1} &= h\,F(t_i + h, \varphi(t_i) + k_i), \\ \varphi(t_{i+1}) &= \varphi(t_i) + \frac{1}{2}(k_i + k_{i+1}). \end{aligned} \tag{28}$$

where $h = L/n$ and $k_0 = hF(t_0, \varphi_0)$. This algorithm requires exactly $n+1$ evaluations of the function F and its order of accuracy is two.

2.7 Bisection and the secant method

The bisection method and the secant method are classical root finding schemes (see, for example, [5]), which we employ for the computation of the stability and accuracy domains. Specifically, the bisection method finds the root of a function f within the interval $[x_1, x_2]$ iteratively, by testing whether f changes its sign to the left or to the right of

$$x_3 = \frac{x_1 + x_2}{2}. \tag{29}$$

If $f(x_3)$ has the same sign as $f(x_2)$ the method sets $x_2 = x_3$ and repeats, else it sets $x_1 = x_3$ and repeats. The bisection stops when x_1 and x_2 are sufficiently close. Evidently, if $[x_1, x_2]$ contains one single root, bisection requires $O(\log(\varepsilon))$ steps to obtain the root with precision ε.

The secant method is an iterative scheme as well, which requires two starting values. It computes the root of the function f by approximating f by its secant through the points x_1 and x_2 which have been obtained during the previous two iterations. The secant through x_1 and x_2 intersects the x-axis at

$$x_3 = \frac{f(x_2)x_1 - f(x_1)x_2}{f(x_2) - f(x_1)}. \tag{30}$$

If $f(x_2) > f(x_1)$ the secant methods sets $x_1 = x_3$ and repeats, else it sets $x_2 = x_3$ and repeats. The order of convergence of the secant method is approximately 1.6, however, the secant method is not guaranteed to converge.

2.8 The maximum modulus principle

The following result can be found in any standard textbook on complex analysis.

Lemma 8 (Maximum modulus principle) *If $D \subset \mathbb{C}$ is a bounded connected set with boundary ∂D, $U \subset \mathbb{C}$ is an open set such that $(D \cup \partial D) \subset U$ and the function $f : U \to \mathbb{C}$ is analytic on U then the maximum value of $|f|$ on $(D \cup \partial D)$ occurs on the boundary, i. e.*

$$\max_{\partial D} |f| = \max_{D} |f|. \tag{31}$$

3 Analytical Apparatus

In this section, we discuss the construction of the extrapolation and integration formulae used by the predictor-corrector methods of this paper (Sections 3.1, 3.1), an efficient representation of functions of bounded frequency (Section 3.3), and spectral methods for functions of bounded frequency on equidistant nodes (Section 3.4).

3.1 The predictor step: Extrapolation of exponentials

The predictor step of a predictor-corrector scheme approximates the solution of the ODE at the next time step by extrapolating the solution based on its values and derivatives at previous time steps. In this section we construct an extrapolation formula, which is based on the assumption that the ODE solution, which is to be extrapolated, is a linear combination of the finite set of exponentials $e^{\lambda_1 t}, e^{\lambda_2 t}, \dots, e^{\lambda_n t}$, where $\lambda_i \in \mathbb{C}$ (The selection of the λ_i is described in Section 3.3 below).

Suppose that the values and derivatives of the functions which are to be extrapolated are given at an equidistant discretization $t_1, t_2, \dots, t_k$ of the interval [-1,1] with step-size h, and they are to be evaluated at $t_{k+1} = 1 + h$ via the formula

$$\varphi(t_{k+1}) = \sum_{i=1}^{k} \left[p_i \varphi(t_i) + p_{k+i} \varphi'(t_i) \right], \tag{32}$$

where $p_1, \dots p_{2k} \in \mathbb{R}$. Formula (32) is valid for all functions φ of the form $\varphi(t) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t}$ with $\alpha_i \in \mathbb{C}$, if $p = (p_1, \dots, p_{2k})^T$ satisfies the linear system

$$Ap = b, \tag{33}$$

where the elements of $A \in \mathbb{C}^{n \times 2k}$ are

$$A_{ij} = \begin{cases} e^{\lambda_i t_j}, & i = 1, \dots, n; j = 1, \dots, k \\ \lambda_i e^{\lambda_i t_{j-k}}, & i = 1, \dots, n; j = k+1, \dots, 2k \end{cases}, \tag{34}$$

and the components of $b \in \mathbb{C}^n$ are

$$b_i = e^{\lambda_i t_{k+1}}. \tag{35}$$

In general, the vector p can be approximated by the minimum norm least squares solution (21) of system (33):

$$p(\varepsilon) = \sum_{i=1}^{r(\varepsilon)} \frac{u_i^* b}{\sigma_i} v_i, \tag{36}$$

where $\varepsilon > 0$ is a fixed precision value, and $r(\varepsilon)$, σ_i, u_i, and v_i are as in equation (21).

Remark 9 The choice of the parameters n, k, and ε, for the construction of the extrapolation formula will affect the stability and accuracy properties of the resulting predictor-corrector scheme. If $2k \geq n$ then the accuracy of the extrapolation formula (32) is of order ε. Furthermore, if n and k are fixed, $||p(\varepsilon)||_2$ tends to grow as ε is decreased, which indicates that increasing ε will improve the stability of the resulting schemes. Similar, increasing the number of steps k will also decrease $||p(\varepsilon)||_2$ and will tend to improve the stability.

We have investigated the stability properties of the predictor-corrector schemes of this paper numerically and will discuss the results in Section 5.

3.2 The corrector step: Integration of exponentials

After using the result for $\varphi(t_{k+1})$ from the predictor step to obtain $\varphi'(t_{k+1})$ via equation (1), the corrector step recomputes the value of $\varphi(t_{k+1})$ via the formula

$$\varphi(t_{k+1}) = \sum_{i=1}^{k} [c_i \varphi(t_i) + c_{k+i} \varphi'(t_i)] + c_{2k+1} \varphi'(t_{k+1}), \tag{37}$$

where $c_1, \ldots, c_{2k+1} \in \mathbb{R}$. As before, we assume that $\varphi(t) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t}$ with $\alpha_i \in \mathbb{C}$. Hence, $c = (c_1, \ldots, c_{2k+1})^T$ has to satisfy

$$\widetilde{A} c = b, \tag{38}$$

where b is as in equation (35), and the elements of $\widetilde{A} \in \mathbb{C}^{n \times 2k+1}$ are

$$\widetilde{A}_{ij} = \left\{ \begin{array}{cl} A_{ij}, & i = 1, \ldots, n; j = 1, \ldots, 2k \\ \lambda_i e^{\lambda_i t_{k+1}}, & i = 1, \ldots, n; j = 2k+1 \end{array} \right. . \tag{39}$$

The vector c can be approximated by the minimum norm least squares solution (21) of system (38).

Remark 10 As in the predictor case, the choice of the parameters n, k and ε will affect the stability and accuracy properties of the resulting predictor-corrector scheme (See Remark 9). In general, the integration during the corrector step is a significantly more stable procedure than the extrapolation during the predictor step; additional corrector steps will therefore typically improve the stability of the resulting predictor-corrector schemes. Section 5 contains a discussion of these stability issues, which is based on numerical experiments.

Remark 11 Both the predictor formula (32) and the corrector formula (37) make use of the values and the derivatives of the function φ. While it would be feasible to base the predictor step and the corrector step only on the derivatives $\varphi'(t_1), \ldots, \varphi'(t_k)$, which is the approach of most classical predictor-corrector schemes, our approach of always using the information about the values and the derivatives leads to significantly more stable formulae.

3.3 Skeletonization of a semi-disk in $\mathbb{C}$

In this section we describe how $\lambda_1, \ldots, \lambda_n$ in equations (34), (39) can be selected such that (32), (37) will be satisfied for all functions $f_\lambda(x) = e^{-\lambda x}$, for which λ lies within the complex semi-disk

$$S_r = \{\lambda \in \mathbb{C} \mid \mathrm{Re}(\lambda) \leq 0,\ |\lambda| \leq r\}\,. \tag{40}$$

The discussion proceeds in two steps. First, Lemma 12 establishes that it is sufficient to discretize the boundary of S_r (denoted by ∂S_r). Then, we construct the discretization of ∂S_r by employing Lemma 6.

Lemma 12 *Given the initial value problem*

$$\begin{aligned} \varphi^{'}(t) &= \lambda\varphi, & (41)\\ \varphi(0) &= \varphi_0, & (42) \end{aligned}$$

let φ_λ denote its solution and $\widetilde{\varphi}_\lambda$ its approximate solution computed via a k-step $PE(CE)^m$ method as described in Section 2.1 (assuming that the k required starting values are given). Furthermore, let D denote a connected bounded region in the complex plane and ∂D its boundary.

If for any fixed $t \in \mathbb{R}$ and $\varepsilon > 0$ the inequality

$$|\varphi_\lambda(t) - \widetilde{\varphi}_\lambda(t)| < \varepsilon \tag{43}$$

holds for all $\lambda \in \partial D$, then (43) also holds for all $\lambda \in D$.

Proof. For any fixed value of t a $PE(CE)^m$ method computes $\widetilde{\varphi}_\lambda(t)$ by a finite number of additions and multiplications of linear functions of λ. The function $g_t : \mathbb{C} \to \mathbb{C}$ defined by $\lambda \mapsto \widetilde{\varphi}_\lambda(t)$ is therefore a polynomial in λ and hence an analytic function of λ. Since the function $f_t : \mathbb{C} \to \mathbb{C}$ defined by $\lambda \mapsto e^{\lambda t}$ is also an analytic function of λ, the error term $(\varphi_\lambda(t) - \widetilde{\varphi}_\lambda(t))$ is an analytic function of λ as well. The lemma follows therefore from the maximum modulus principle 8. $\square$

Lemma 12 implies that it is sufficient to construct a discretization the boundary of S_r, which we accomplish as follows. Since the result of one step of a $PE(CE)^m$ method depends linearly on the solution at the previous nodes, any method valid for the functions $e^{\lambda_1 t}, \ldots, e^{\lambda_n t}$ will also be valid for any linear combination of these functions. In particular, let $e^{\lambda_1 t}, \ldots, e^{\lambda_n t}$ be a collection of functions such that for a

δ	10^{-5}	10^{-7}	10^{-10}	10^{12}	10^{-16}	10^{-20}	10^{-30}
n	4	12	18	24	30	38	54

Table 1: Number n of nodes for different accuracies δ (Skeletonization of $S_{3.15}$)

specified precision δ and for any $\lambda \in \partial S_r$ there exist coefficients $c_1, \ldots, c_n \in \mathbb{C}$ such that

$$\left| e^{\lambda t} - \sum_{i=1}^{n} c_i e^{\lambda_i t} \right| < \delta, \tag{44}$$

for all $t \in [-1, 1]$. If (43) is satisfied for $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$, then

$$|\varphi_\lambda(t) - \widetilde{\varphi}_\lambda(t)| < \varepsilon + \delta, \tag{45}$$

for all $\lambda \in \partial S_r$ (triangle inequality). Therefore, it is sufficient to select a set of exponentials which can generate all exponentials $\left\{ e^{\lambda t} \middle| \, \lambda \in \partial S_r \right\}$ up to a finite precision δ via linear interpolation. Since exponentials of the form $e^{\lambda t}$ are smooth functions of λ, it is clear that such a selection of exponentials exists; a specific example is a sufficiently dense equidistant discretization of ∂S_r, where the arclength between two neighboring nodes in the discretization is the same for all neighboring pairs.

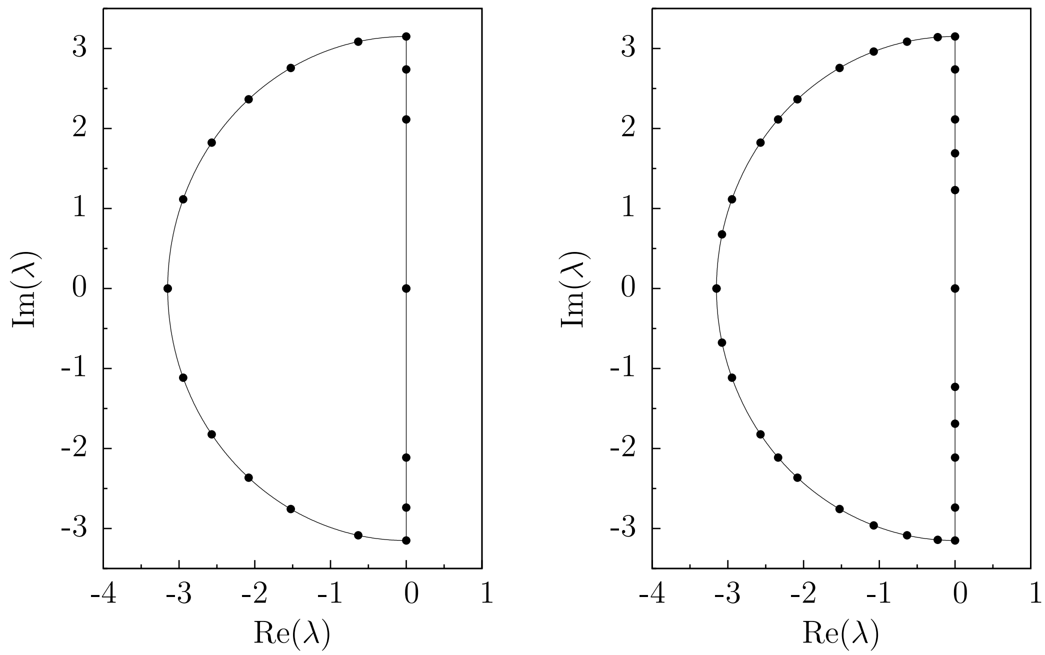
Lemma 6 provides a tool to choose an efficient discretization of the type discussed above. Starting with an equidistant discretization, we employ the factorization in Lemma 6 to significantly reduce the number of required nodes. Suppose that $\lambda_1, \ldots, \lambda_N$ and $t_1, \ldots, t_M$ are sufficiently dense equidistant discretizations of ∂S_r and [-1,1], respectively, and the factorization (22) of

$$A = \begin{pmatrix} e^{\lambda_1 t_1} & e^{\lambda_2 t_1} & \cdots & e^{\lambda_N t_1} \\ e^{\lambda_1 t_2} & e^{\lambda_2 t_2} & \cdots & e^{\lambda_N t_2} \\ \vdots & \vdots & & \vdots \\ e^{\lambda_1 t_M} & e^{\lambda_2 t_M} & \cdots & e^{\lambda_N t_M} \end{pmatrix}, \tag{46}$$

with error term $||X||_2 < \delta$ yields the k columns with indices $\{i_1, \ldots, i_k\}$. Then the exponentials $e^{\lambda_{i_1} t}, \ldots, e^{\lambda_{i_k} t}$ span $\left\{ e^{\lambda t} \middle| \, \lambda \in \partial S_r \right\}$ up to precision δ and are therefore an appropriate representation of $\left\{ e^{\lambda t} \middle| \, \lambda \in \partial S_r \right\}$ (and hence of $\left\{ e^{\lambda t} \middle| \, \lambda \in S_r \right\}$), when the precision δ is required.

For the values of δ and r which are of interest in this paper, the above procedure, often referred to as skeletonization, results in a number of nodes between 12 and 54, as illustrated in Table 1, which shows the number of nodes n that are required for the skeletonization of S_r with $r = 3.15$ for different accuracies δ. Figure 1 shows the skeletonization of $S_{3.15}$ for the precisions $\delta = 10^{-10}$ and $\delta = 10^{-16}$.

Remark 13 Since ∂S_r is symmetric with respect to the real axis, it is sufficient to discretize the upper half of ∂S_r ($\mathrm{Im}(\lambda) \geq 0$) and choose the discretization of the lower half of ∂S_r, such that it is symmetric to the upper half. In order to ensure that this leads to an efficient discretization close to the real axis, the discretization of the upper half of ∂S_r can be constrained to include $\lambda_1 = 0$ and $\lambda_2 = -r\sqrt{-1}$.



(a) $\delta = 10^{-10}$ (b) $\delta = 10^{-16}$

Figure 1: Skeletonization of S_r with $r = 3.15$

Remark 14 The construction of predictor and corrector formulae (32), (37) makes use of the assumption that the numerical solution to equation (6) is sampled at an equidistant discretization $t_1, \ldots, t_k$ of [-1,1] with step-size $h = 2/(k-1)$ and is of the form $f_\lambda = e^{\lambda t}$, where $\lambda \in S_r$. If the resulting formulae are applied to a different equidistant sequence of nodes $\tilde{t}_1, \ldots, \tilde{t}_k$ with the new step-size $\widetilde{h} = \alpha h$, they will be valid for all solutions of the form $f_\lambda = e^{\lambda t}$, where $\lambda \in S_{r/\alpha}$. In other words, halving the step-size will double the maximal frequency of the functions to which the predictor-corrector scheme is applicable. This follows from the fact that (1) is invariant under translations and, furthermore, the dilation $\tau = \alpha\ t$ transforms equation (6) into

$$\frac{\partial}{\partial \tau}\varphi(\tau) = \frac{\lambda}{\alpha}\varphi(\tau). \tag{47}$$

Remark 15 The choice of the parameter r in (40) determines the maximal frequency of the functions to which the predictor and corrector formulae (32), (37) apply for the fixed step-size $h = 2/k$. By construction, the *number of steps per wavelength* to achieve the optimal accuracy of the resulting scheme will therefore be approximately $\frac{k\pi}{r}$.

3.4 Spectral deferred correction on equidistant nodes

In this section we describe a spectral deferred correction method to be used to compute the starting values for the predictor-corrector scheme. The deferred correction method will be constructed on the equidistant discretization $t_1, t_2, \ldots, t_k$ of $[-1, 1]$ with step-size h. Furthermore, we will assume that the desired solution to (1), (2) is a linear combination of $e^{\lambda_1 t}, e^{\lambda_2 t}, \ldots, e^{\lambda_n t}$, where $\lambda_1, \ldots, \lambda_n$ is a discretization of ∂S_r as discussed in the previous section.

Given these assumptions, the value for $\varepsilon(t_i)$ as defined in (24) can be evaluated by the quadrature rule

$$\int_{-1}^{t_j} \varphi(\tau)\tau d\tau \approx \sum_{i=1}^{k} \varphi(t_i) r_{ij}, \tag{48}$$

where $r_{ij} \in \mathbb{R}$. The corresponding weight matrix $r = (r_{ij})$ has to satisfy the linear system

$$Ar = b, \tag{49}$$

where $A \in \mathbb{C}^{n \times k}$,

$$A = \begin{pmatrix} e^{\lambda_1 t_1} & e^{\lambda_1 t_2} & \cdots & e^{\lambda_1 t_k} \\ e^{\lambda_2 t_1} & e^{\lambda_2 t_2} & \cdots & e^{\lambda_2 t_k} \\ \vdots & \vdots & & \vdots \\ e^{\lambda_n t_1} & e^{\lambda_n t_2} & \cdots & e^{\lambda_n t_k} \end{pmatrix}, \tag{50}$$

and $b \in \mathbb{C}^{n \times k}$,

$$b = \begin{pmatrix} \int_{-1}^{t_1} e^{\lambda_1 \tau} d\tau & \cdots & \int_{-1}^{t_k} e^{\lambda_1 \tau} d\tau \\ \int_{-1}^{t_1} e^{\lambda_2 \tau} d\tau & \cdots & \int_{-1}^{t_k} e^{\lambda_2 \tau} d\tau \\ \vdots & & \vdots \\ \int_{-1}^{t_1} e^{\lambda_n \tau} d\tau & \cdots & \int_{-1}^{t_k} e^{\lambda_n \tau} d\tau \end{pmatrix}. \tag{51}$$

Approximating each column of the matrix r by the minimum norm least squares solution (21) of precision ε of (49) results in a quadrature formula whose accuracy is of order ε. Since the integral and its approximation are analytical functions of λ, it follows from the maximum principle (Lemma 8) that (48) also holds for all linear combinations of $\{e^{\lambda t} \mid \lambda \in S_r\}$ (see Lemma 12).

Equations (25) and (27) can be solved by applying a second order Runge-Kutta scheme (28). In particular, the second order Runge-Kutta scheme for the solution of equation (27) with step-size h is given by

$$\begin{aligned} \widetilde{\gamma}_{i+1} &= \gamma_i + hG(t_i, \widetilde{\gamma}_i) + \varepsilon(t_{i+1}) - \varepsilon(t_i), \\ \gamma_{i+1} &= \gamma_i + \frac{h}{2}\left[G(t_i, \widetilde{\gamma}_i) + G(t_i, \widetilde{\gamma}_{i+1})\right] + \varepsilon(t_{i+1}) - \varepsilon(t_i), \end{aligned} \tag{52}$$

where $G(t, \gamma) = F(t, \varphi^j(t) + \gamma) - F(t, \varphi^j(t))$ and γ_i denotes $\gamma(t_i)$.

Remark 16 The resulting deferred correction scheme makes use of the assumption that the numerical solution to equation (1) is obtained at an equidistant discretization $t_1, \ldots, t_k$ of [-1,1] with step-size $h = 2/(k-1)$ and is of the form $f_\lambda = e^{\lambda t}$, where $\lambda \in S_r$. If the scheme is applied to on the nodes $\tilde{t}_1, \ldots, \tilde{t}_k$ with the new step-size $\widetilde{h} = \alpha h$, it will be valid for all solutions of the form $f_\lambda = e^{\lambda t}$, where $\lambda \in S_{r/\alpha}$ (see Remark 14).

Remark 17 The stability and accuracy properties of the deferred correction schemes discussed in this section have been established numerically and are described in Section 5.

4 Description of the Algorithm

This section contains a description of the algorithms that are the principal goal of this paper. The informal description in the section below is followed by a more detailed description in Section 4.2.

4.1 Informal description

Several precomputation steps of the algorithm require a discretization of S_r as defined in (40), which can be precomputed via factorization (22) of matrix (46). As a first step the algorithm uses a spectral deferred correction scheme to compute the solution to the initial value problem (1), (2) at k equidistant nodes with step-size h, which are used as the starting values for the predictor-corrector scheme. The initial solution for the deferred correction iterations is obtained via a second order Runge-Kutta scheme (28), after which the solution is improved iteratively via (24), (26), (27), where $\gamma(t)$ is computed via the Runge-Kutta scheme (52) and the integral in (24) is evaluated via the approximation (48). The integration vectors $r_1, \ldots, r_k$ are the minimum norm least squares solution (21) of (49), and can be precomputed. The iterations stop when at all k nodes, $\gamma(t)$ in equation (27) is less than a pre-set accuracy ε.

After the initial k values of the solution $\varphi(t)$ have been computed, the algorithm advances the solution, taking steps of size h, via a k-step $PE(CE)^m$ method (see Section 2.1). During each time step one predictor step is followed by m corrector steps. The predictor and corrector steps are based on the assumption that the solution to (1) can be represented by a linear combination of exponentials $e^{\lambda}t$, where λ lies in the complex semi-disk S_r (40). In particular, the predictor step approximates $\varphi(t_{k+1})$ via (32), where the vector p in (32) is the minimum norm least squares solution of (33) and can is precomputed once and for all.

After the predictor step, the approximate value of $\varphi(t_{k+1})$ is used to compute $\varphi'(t_{k+1})$ via (1). The corrector step then re-computes $\varphi(t_{k+1})$ via formula (37), where the vector c in (37) is the minimum norm least squares solution of (38) and is precomputed once and for all. The corrector step is repeated m times.

If the interval $[a, b]$ is discretized into N points then the predictor corrector scheme requires $(m+1) \cdot N$ evaluations of the function F in (1), where m is number of times the corrector step is applied after each predictor step.

Remark 18 If (1), (2) is a system of ODEs, i. e. the dimension of $\varphi(t)$ is bigger than one, the same algorithm can be applied by performing all steps componentwise.

4.2 Detailed description

In this section, we describe the algorithm of this paper in some detail. The presentation proceeds in two parts: Algorithm 1.1 is the procedure for computing the starting values for the predictor-corrector method via a spectral deferred correction scheme based on equidistant nodes; Algorithm 1.2 is the main algorithm, which is a k-step $PE(CE)^m$ method.

Algorithm 1.1 (Spectral Deferred Correction)
Comment [Algorithm 1.1 computes the solution to initial value problem (1), (2) at a given equidistant discretization $t_1, \ldots, t_k$ via a spectral deferred correction scheme. It has two parts: the precomputation which is performed offline and the main algorithm. Its implementation requires the choice of the following parameters: k is the number of nodes in the discretization, ε_{LS}^D,δ,M,N are parameters during the precomputation, which will determine the accuracy properties of the algorithm, and finally ε_I is the stopping criterion for the correction iterations.]

Precomputation

1. Compute a discretization $\lambda_1, \ldots, \lambda_n$ of S_r by computing the factorization (22) of the $M \times N$ matrix (46), such that $||X||_2 < \delta$.

2. Compute r as the minimum norm least squares solution of precision ε_{LS}^D of the system $Ar = b$, where A and b_i are defined in (50), (51).

Main algorithm

Compute $\varphi^0(t_1), \ldots, \varphi^0(t_k)$ by solving equation (1) with initial condition $\varphi(t_1) = \varphi_0$ via the second-order Runge-Kutta method (28).

Repeat

1. For $j = 1, \ldots, n$, compute

$$\varepsilon(t_j) = \varphi^j(t_1) + \sum_{i=1}^{k} F(t_i, \varphi^j(t_i)) r_{ij} - \varphi^j(t_j), \tag{53}$$

   where the matrix $r = (r_{ij})$ has been precomputed.

2. Compute $\gamma(t)$ at $t_1, \ldots, t_k$ by solving equation (27) with the initial condition $\gamma(t_1) = 0$ via the Runge-Kutta scheme (52).

3. Set $\varphi^{j+1}(t_i) = \varphi^j(t_i) + \gamma(t_i)$ for $t_1, \ldots, t_k$.

until $\gamma(t_i) < \varepsilon_I$ for $i = 1, \ldots, k$

Return $\varphi^{j+1}(t_1), \ldots, \varphi^{j+1}(t_k)$.

Algorithm 1.2 (Predictor-Corrector)
Comment [Algorithm 1.2 computes the solution to initial value problem (1), (2) at a given equidistant discretization $t_1, \dots, t_N$ via a k-step $PE(CE)^m$ method. It requires the starting values $\varphi(t_1), \dots, \varphi(t_k)$ as input and has two parts: the precomputation and the main algorithm. The precomputation requires the choice of the parameters δ, M, N,ε_{LS}^P, ε_{LS}^C, which will be introduced below.]

Precomputation

1. Compute a discretization $\lambda_1, \dots, \lambda_n$ of S_r by computing the factorization (22) of the $M \times N$ matrix (46), such that $||X||_2 < \delta$.
2. Compute the minimum norm least squares solution (21) of precision ε_{LS}^P of the system $Ap = b$, where A and b are defined in (34) and (35), respectively.
3. Compute the minimum norm least squares solution (21) of precision ε_{LS}^C of the system $\widetilde{A}c = b$, where A and b are defined in (39) and (35), respectively.

Main Algorithm

do $j = k, N-1$

1. Compute $\varphi(t_{j+1})$ via
$$\varphi(t_{j+1}) = \sum_{i=1}^{k} \left[p_i \varphi(t_{(j-k)+i}) + p_{k+i} \varphi'(t_{(j-k)+i}) \right], \tag{54}$$
where the vector $p = (p_1, \dots, p_{2k})$ has been precomputed.
2. Compute $\varphi'(t_{j+1}) = F\left(t_{j+1}, \varphi(t_{j+1})\right)$.
3. Recompute $\varphi(t_{j+1})$ via
$$\varphi(t_{j+1}) = \sum_{i=1}^{k} \left[c_i \varphi(t_{(j-k)+i}) + c_{k+i} \varphi'(t_{(j-k)+i}) \right] + c_{2k+1} \varphi'(t_{j+1}), \tag{55}$$
where the vector $c = (c_1, \dots, c_{2k+1})$ has been precomputed. Then recompute $\varphi'(t_{j+1})$ as in the previous step. Perform this step m times.

end do

Return $\varphi(t_1), \dots, \varphi(t_N)$.

Name	r	M	N	δ	n	k	ε_D	ε_I
DC1	3.15	800	800	10^{-10}	18	22	10^{-9}	10^{-10}
DC2	6.3	800	800	10^{-19}	46	60	10^{-18}	10^{-15}
DC3	3.15	800	800	10^{-19}	38	42	10^{-19}	10^{-15}
DC4	6.3	800	800	10^{-36}	76	80	10^{-32}	10^{-31}

Table 2: Implementations of the deferred correction scheme (Algorithm 1.1)

Name	r	M	N	δ	n	k	ε_P	ε_C
PC1	3.15	800	800	10^{-10}	18	22	10^{-9}	10^{-9}
PC2	6.3	800	800	10^{-17}	46	60	10^{-16}	10^{-16}
PC3	3.15	800	800	10^{-20}	38	42	10^{-19}	10^{-18}
PC4	3.15	800	800	10^{-34}	76	80	10^{-30}	10^{-32}

Table 3: Implementations of the predictor-corrector scheme (Algorithm 1.2)

5 Numerical Experiments

We have implemented the spectral deferred correction scheme (Algorithm 1.1) and the predictor-corrector method (Algorithm 1.2) for different sets of parameters and tested their performance. This section discusses the results in three steps. First, it gives a detailed description of the implemented schemes. Second, it describes the numerical experiments establishing the schemes' stability and accuracy properties. And, finally, the performance of the implemented schemes is demonstrated on two examples.

The implementation of Algorithm 1.1 and Algorithm 1.2 requires several parameter choices. Both, Algorithm 1.1 and Algorithm 1.2, require a discretization of the complex semi-disk S_r (40). The discretization depends on the radius of S_r, denoted by r, and on the number of rows and columns of matrix A in (46), denoted by M and N, respectively. Furthermore, it depends on the accuracy of the factorization (22), i.e. the operator norm of X in (22), of matrix A (46), which will be denoted by δ. Finally, the number of points in the resulting discretization will be denoted by n. The parameters for the computation of the discretization of S_r employed in the implementations discussed in this section are listed in columns $3-7$ of Table 2 and Table 3.

Remark 19 In all cases, the parameters M and N (the number of columns and rows in the matrix A in (46)) have been set to 800, and the calculations were performed in extended precision (32 digits). This number was determined empirically for the 31-digit case; since it is guaranteed to be sufficient for all lower accuracies and results in acceptable precomputation times, no further analysis has been performed.

The deferred correction scheme (Algorithm 1.1) computes the integration matrix r in equation (49) by a minimum norm least squares fit (21). The accuracy of this fit, i.e. ε in (21), will be denoted by ε_D and the number of time steps (the number of columns of r) will be denoted by k. The computation of r is a precomputation and has been performed in extended precision (32-digits). The ε in the stopping condition (Step 4) of Algorithm 1.1 is denoted by ε_I. In our implementation we perform one more iteration once this accuracy has been achieved. This section discusses four different implementations of the deferred correction scheme. The parameters of each implementation are listed in Table 2.

The predictor-corrector scheme precomputes the predictor and corrector formulae by calculating the minimum norm least squares solution (21) of equations (33) and (38). The number of time steps in these formulae is denoted by k. The accuracies of minimum norm least squares solutions, i.e. ε in (21), in the predictor and corrector case are denoted by ε_P and ε_C, respectively. Again, the precomputations have been performed in extended precision (32 digits). The parameters of the implementations of the predictor corrector scheme discussed in this section are listed in Table 3. Furthermore, as an example, Table 4 lists the integration and extrapolation vectors p and c of the scheme PC1, which are obtained during Step 5 of the precomputations of Algorithm 1.2.

All schemes have been implemented in FORTRAN using the Lahey-Fujitsu FORTRAN 95 compiler with the optimization flag set to -o2. Furthermore, the timings in Section 5.2 have been measured on an Intel Pentium 4, 3.2 GHz with 2 GB RAM.

5.1 Stability and accuracy properties

This section discusses the stability and accuracy properties of the deferred correction implementations listed in Table 2 and of the predictor-corrector implementations listed in Table 3. Specifically, we numerically construct the stability and accuracy domains of the implemented schemes, which are defined as follows (see also, Section 2.2).

Let $\widetilde{\varphi}$ denote the numerical solution to equations (6), (7) obtained at an equidistant discretization of $[0,1]$ with step-size h via one of the deferred correction schemes listed in Table 2. The stability domain of the scheme is the set of all values λh for which $|\widetilde{\varphi}(1)| \leq 1$. Furthermore, for a given ε the accuracy domain is defined to be the set of all λh, for which

$$\sqrt{\frac{\sum_{i=1}^{k} |\varphi(t_i) - \widetilde{\varphi}(t_i)|^2}{\sum_{i=1}^{k} |\varphi(t_i)|^2}} < \varepsilon, \tag{56}$$

where $\varphi : \mathbb{C} \mapsto \mathbb{C}$, $z \mapsto e^{\lambda z}$ denotes the analytical solution to (6), (7).

Similar, let $\widetilde{\varphi}$ denote the numerical solution to (6), (7) obtained at an equidistant discretization $t_1, t_2, \ldots, t_n$ of $[0,1]$ of step-size h via one of the predictor corrector schemes listed in Table 3 with the given starting values $\widetilde{\varphi}(t_1), \ldots, \widetilde{\varphi}(t_k)$. The stability domain of the scheme is the set of all values λh for which $|\widetilde{\varphi}(t_i)| \leq 1$ holds for all $i = 1, \ldots, n$. The latter condition is satisfied if the largest singular value of the matrix B defined in equation (10) is less than one.

	p	c
1	$5.6950867745996450 \times 10^{-2}$	$2.2410253811826690 \times 10^{-2}$
2	$1.2471828303461370 \times 10^{-2}$	$2.4633154278189170 \times 10^{-2}$
3	$2.8894215936093530 \times 10^{-2}$	$2.4336066393818950 \times 10^{-2}$
4	$2.4083554081417720 \times 10^{-2}$	$2.5149950648957660 \times 10^{-2}$
5	$1.7345252832690130 \times 10^{-2}$	$2.6486814374770810 \times 10^{-2}$
6	$2.8519730870305910 \times 10^{-2}$	$2.7234737421409750 \times 10^{-2}$
7	$3.6071920983279110 \times 10^{-2}$	$2.8198076103084640 \times 10^{-2}$
8	$2.8966177271448650 \times 10^{-2}$	$3.0096942089726130 \times 10^{-2}$
9	$2.2315178842523700 \times 10^{-2}$	$3.2486068591206280 \times 10^{-2}$
10	$2.9905186291956070 \times 10^{-2}$	$3.4641908350425040 \times 10^{-2}$
11	$4.4440048810368700 \times 10^{-2}$	$3.6616005690985700 \times 10^{-2}$
12	$4.9069767898691480 \times 10^{-2}$	$3.9130905383232570 \times 10^{-2}$
13	$4.0650298910123330 \times 10^{-2}$	$4.2655375218158010 \times 10^{-2}$
14	$3.4669719074163200 \times 10^{-2}$	$4.6832060662658940 \times 10^{-2}$
15	$4.5611440916694790 \times 10^{-2}$	$5.0917965571713190 \times 10^{-2}$
16	$6.6034586913433500 \times 10^{-2}$	$5.4758986719566050 \times 10^{-2}$
17	$7.2628793760997530 \times 10^{-2}$	$5.9165803238876420 \times 10^{-2}$
18	$5.9216748674729610 \times 10^{-2}$	$6.5051564452405580 \times 10^{-2}$
19	$5.6406142035183890 \times 10^{-2}$	$7.2079207171644550 \times 10^{-2}$
20	$8.9720539463978590 \times 10^{-2}$	$7.8700198716440890 \times 10^{-2}$
21	$1.0201521115897360 \times 10^{-1}$	$8.4738392656218300 \times 10^{-2}$
22	$5.4012787919006680 \times 10^{-2}$	$9.3679562525744830 \times 10^{-2}$
23	$6.0560922579890260 \times 10^{-3}$	$3.7583455331903450 \times 10^{-4}$
24	$-2.7807762994373060 \times 10^{-2}$	$5.8682877788308760 \times 10^{-3}$
25	$1.1956698573336000 \times 10^{-1}$	$-7.0838293847688030 \times 10^{-4}$
26	$-1.4368259417051200 \times 10^{-1}$	$1.5022356394256900 \times 10^{-2}$
27	$4.0276974389784590 \times 10^{-2}$	$1.0788479881224430 \times 10^{-2}$
28	$1.4372084163410750 \times 10^{-1}$	$6.8440197117482410 \times 10^{-3}$
29	$-6.8513418990535620 \times 10^{-3}$	$1.4847045428708830 \times 10^{-2}$
30	$-1.0944090055846840 \times 10^{-1}$	$2.4033339162535270 \times 10^{-2}$
31	$-2.0192137915885270 \times 10^{-3}$	$2.4650259312919750 \times 10^{-2}$
32	$1.4595770836923140 \times 10^{-1}$	$2.0743990881309220 \times 10^{-2}$
33	$1.2479716863929200 \times 10^{-1}$	$2.2174318623909730 \times 10^{-2}$
34	$-3.4116614768704690 \times 10^{-2}$	$3.1578482298307720 \times 10^{-2}$
35	$-1.1144883327854800 \times 10^{-1}$	$4.1718536023450730 \times 10^{-2}$
36	$1.5733708304525340 \times 10^{-2}$	$4.4487205746717590 \times 10^{-2}$
37	$2.0002981731434640 \times 10^{-1}$	$4.0960879667910390 \times 10^{-2}$
38	$1.8114967731355230 \times 10^{-1}$	$4.1289524931745020 \times 10^{-2}$
39	$-5.8479574699884930 \times 10^{-2}$	$5.3143311356800410 \times 10^{-2}$
40	$-1.6214192347450070 \times 10^{-1}$	$6.9904178604996720 \times 10^{-2}$
41	$1.6621388638955530 \times 10^{-1}$	$7.4460084585029470 \times 10^{-2}$
42	$4.2830160005071360 \times 10^{-1}$	$6.3619818896987330 \times 10^{-2}$
43	$-3.5531444249916690 \times 10^{-1}$	$7.0261382310043240 \times 10^{-2}$
44	$3.1221966556634160 \times 10^{-1}$	$1.2124592734310950 \times 10^{-1}$
45	-	$3.1145960381730385 \times 10^{-2}$

Table 4: The vectors p and c for the scheme PC1

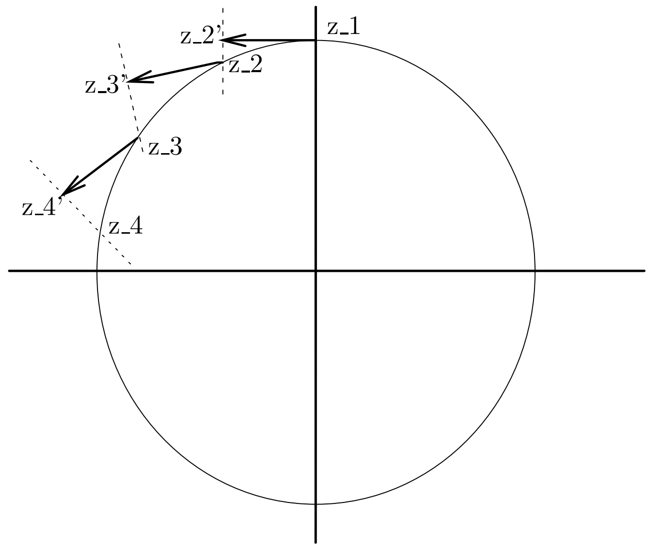


Figure 2: Stepping scheme for the stability and accuracy domain search

For a given ε, the accuracy domain of the predictor corrector schemes listed in Table 3 is defined to be the set of all λh, for which

$$\sqrt{\frac{\sum_{i=k+1}^{n} |\varphi(t_i) - \widetilde{\varphi}(t_i)|^2}{\sum_{i=k+1}^{n} |\varphi(t_i)|^2}} < \varepsilon, \tag{57}$$

where $\varphi : \mathbb{C} \mapsto \mathbb{C}$, $z \mapsto e^{\lambda z}$ denotes the analytical solution to equations (6), (7).

Since the numerical solution $\widetilde{\varphi}$ to (6), (7) is an analytical function of λ for both, the deferred correction and the predictor corrector schemes, and furthermore, the analytical solution φ to (6), (7) is an analytical function of λ as well, it follows from the maximum principle (Lemma 8) that the stability and accuracy domains have well-defined boundaries. In the case of the stability domain the boundary is the set of points for which $|\varphi(1)|$ equals one. In the case of the accuracy domain the boundary is the set of points for which the lefthand sides of (56), (57) equals ε.

The numerical construction of the boundaries is illustrated in Figure 2. First, the secant method (30) is used to search for the boundary point $z_1 \in \mathbb{C}$ along the imaginary axis in the upper half plane. If the secant method fails to converge, z_1 is computed by bisection (29). Second, a small step is taken from z_1 into the negative half plane along a line orthogonal to the imaginary axis ending at the point z_2'. The secant method is used to search for the next boundary point z_2 within a small interval around z_2' along the line which passes through z_2' and is orthogonal to the line through z_1 and z_2' . As before, if the secant method fails, z_2 is computed by bisection. Next, a small step is taken starting from z_2 along the vector from z_1 to z_2 and ending at z_3'. The whole procedure is repeated until a step crosses the imaginary axis in the upper half plane.

All stability domains were computed in double precision (16 digits). The accuracy domains were computed in double precision (16 digits), if $\varepsilon > 10^{-12}$, and in extended precision (32 digits) for the cases of smaller ε.

Figure 3 shows the stability domains and the accuracy domains for various ε for the deferred correction schemes listed in Table 2, while Figures 4 and 5 show the domains of the predictor-corrector schemes listed in Table 3. In each figure the accuracy domains are marked by the label "$\varepsilon = x$", where x stands for the specific choice of ε for the domain in question. The stability domains are indicated by the label "S". For each predictor-corrector scheme the stability domains are shown for the cases of one, two and three correction steps, labeled by "$m = 1$", "$m = 2$" and "$m = 3$", respectively.

The presented figures and our more detailed numerical experiments motivate the following observations.

1. The specific choice of the discretization parameters has only a minor effect on the properties of the resulting deferred correction or predictor-corrector scheme, as long as M and N are sufficiently large (in our experiments 800) and δ is smaller than the desired accuracy of the scheme.

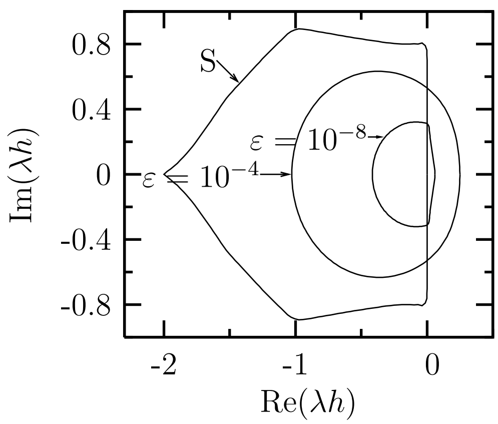
2. The stability domains of the *deferred correction schemes* are quite insensitive to changes in the parameters k, ε_D and ε_I and, hence, the stability domains of the schemes DC1-DC4 are quite similar.

3. The accuracy domain for a deferred correction scheme tends to be close to (but slightly smaller than) the one prescribed. For example, in the case of DC1 with $r = 3.15$, $k = 22$ the scheme is constructed to obtain the precision $\varepsilon_D = 10^{-9}$ within a semi-circle in the λh-plane with radius 0.287 (since during the construction $h = 2/k \approx 0.091$). Figure 3(a) shows, that as, a matter of fact, DC1 obtains the precision 10^{-8} within an approximate semi-circle of radius 0.28, which corresponds to 22 steps per wavelength.

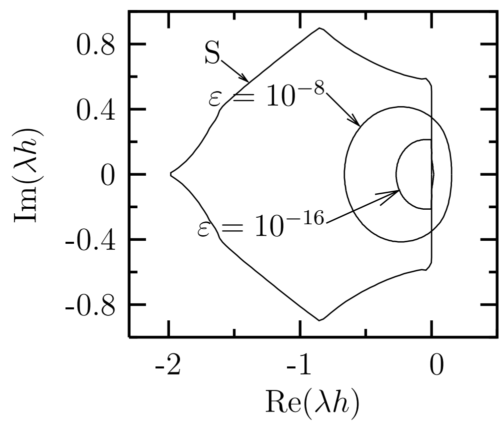
   Consequently, it is straightforward to construct spectral deferred correction schemes, which obtain a desired precision. The schemes we implemented produce 8 digits at 22 steps per wavelength (DC1), 16 digits at 30 steps per wavelength (DC2), 15 digits at 30 steps per wavelength (DC3) and 28 digits at 42 steps per wavelength (DC4).

4. As expected, in the case of the *predictor-corrector schemes* the corrector step is significantly more stable than the predictor step. Hence, the stability domains of the predictor-corrector schemes increase with the number of correction steps m. Overall, the stability domains of the predictor-corrector schemes are very sensitive to the choice of k, ε_P and ε_C. Increasing k, while fixing ε_P and ε_C will increase the stability domain. Consequently, a straightforward procedure for choosing k is to fix ε_P and ε_C at the desired precision and to increase k until the stability domain with $m = 1$ encompasses the accuracy domain of the desired ε.

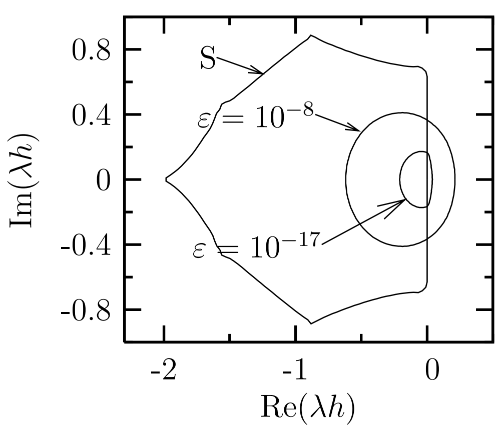
5. Similar to the spectral deferred correction schemes, the accuracy domains of the predictor-corrector schemes is close to the one prescribed, making it rather



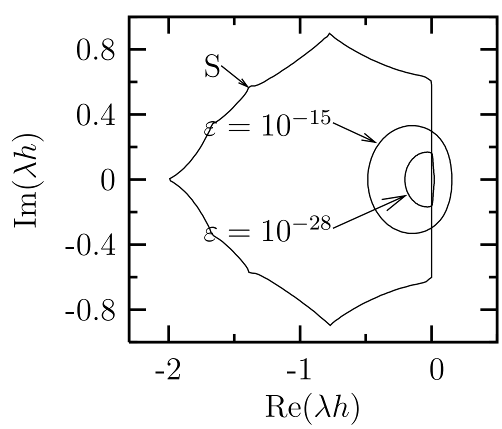
(a) DC1



(b) DC2



(c) DC3



(d) DC4

Figure 3: Stability and accuracy domains of the deferred correction schemes

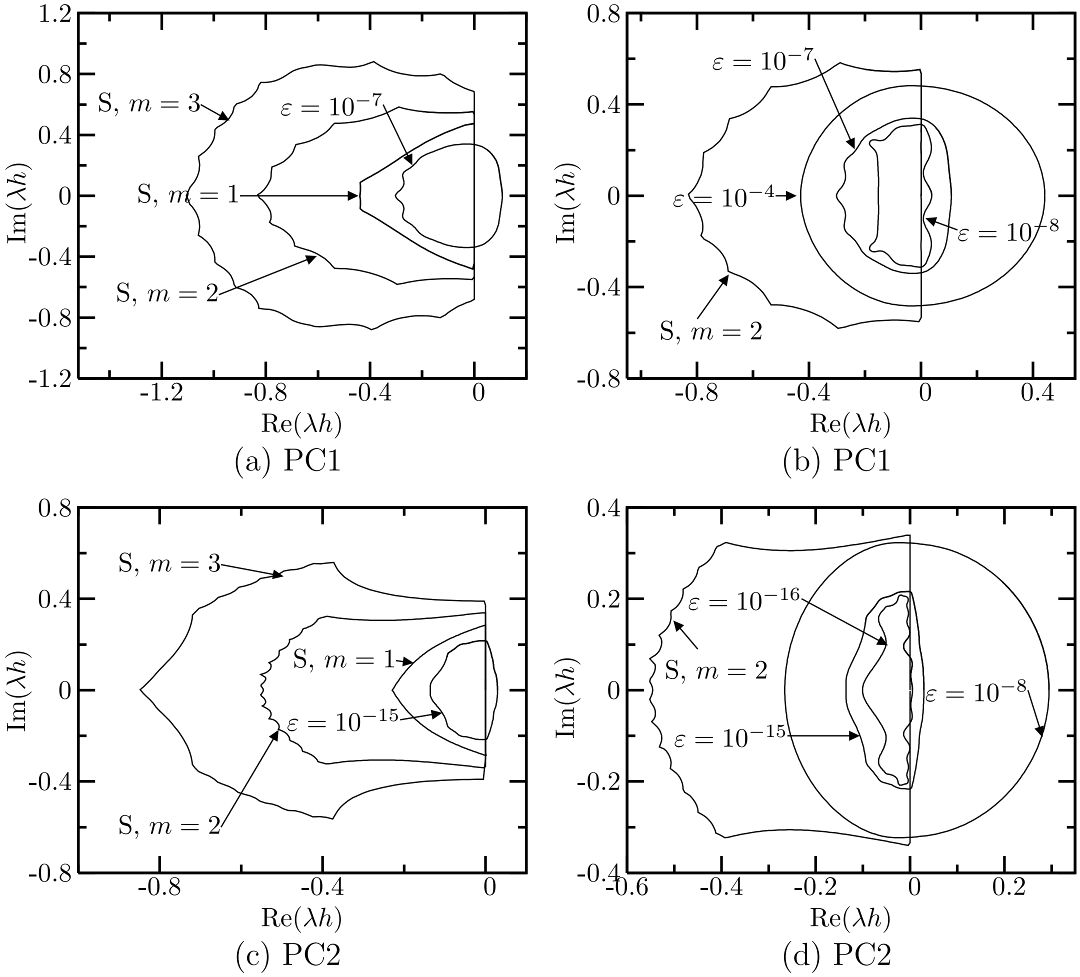


Figure 4: Stability and accuracy domains of PC1 and PC2

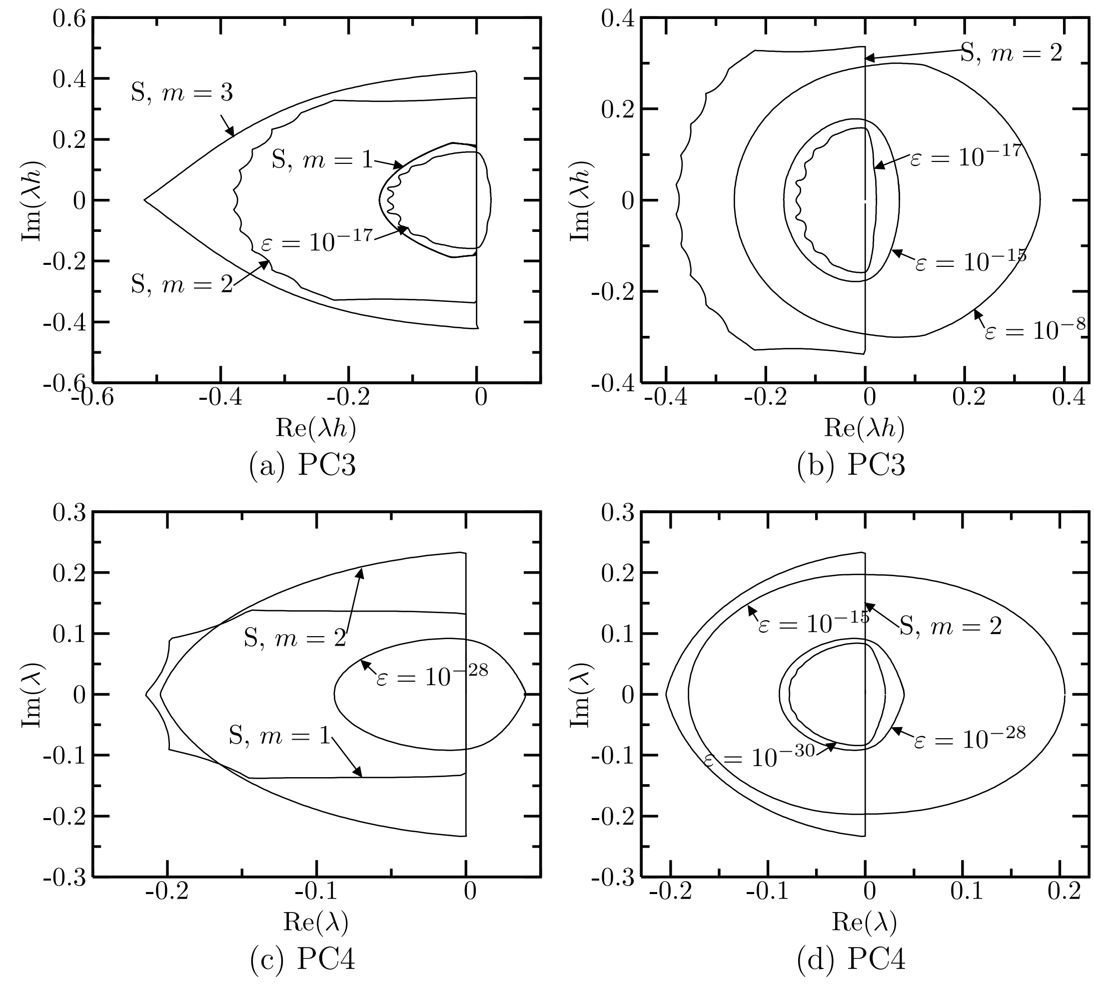


Figure 5: Stability and accuracy domains of PC3 and PC4

straightforward to construct schemes with a specific precision. The schemes we implemented produce 7 digits at 22 steps per wavelength (PC1), 15 digits at 38 steps per wavelength (PC2), 15 digits at 36 steps per wavelength (PC3) and 28 digits at 66 steps per wavelength.

6. Outside the region where a deferred correction scheme or a predictor-corrector scheme achieves its maximal accuracy, the obtained precision decays slowly. For example, at 21 steps per wavelength the scheme DC4 still obtains 15 digits.

5.2 Numerical examples

In this section we illustrate the performance of the presented schemes on two standard examples for non-stiff problems: the Bessel differential equation and the Jacobi elliptic functions.

For both examples of this section we have used each of the four predictor-corrector schemes listed in Table 3 with one corrector step per time step to obtain a numerical solution at various equidistant discretizations. Each of these schemes requires k (see Column 7 of Table 3) starting values, which are computed via the corresponding deferred correction schemes listed in Table 2. Computations have been performed in double (16-digit) precision (schemes DC1+PC1, DC2+PC2 and DC3+PC3), as well as in extended (32-digit) precision (schemes DC2+PC2 and DC4+PC4). For each example we have computed several measures of performance (see Tables 5-12) and compared them to other numerical ODE solvers (see Figures 8-10 and 13-15).

The benchmark methods we use as a comparison are highly optimized Mathematica implementations of a Runge-Kutta scheme and an Adams-Bashforth-Moulton predictor-corrector method. The Runge-Kutta scheme corresponds to the Mathematica function "NDSolve" with the option "Method→ExplicitRungeKutta". It adaptively changes the order of the Runge-Kutta method between 1 and 8. The Adams-Bashforth-Moulton scheme corresponds to the function "NDSolve" with the option "Method→Adams". It changes the order of accuracy adaptively between 1 and 12. Both methods adaptively control the step-size to achieve a specified accuracy, i.e. the discretization nodes at which the solution is returned depend on the specific problem.

Remark 20 We have chosen the Mathematica routines because their performance appears to be representative of state-of-the-art direct ODE solvers. It should be noted, however, that the additional layers of order and step-size control of the Mathematica routines, as well as the different compiler environment, make a direct comparison of the underlying algorithms problematic. The performance comparison should therefore be seen more as an indication than as an absolute measure of the relative performance.

5.2.1 Bessel differential equation

The Bessel function J_n of order n is analytic in the whole plane and satisfies the differential equation (see, for example, [1])

$$x^2 J_n''(x) + x\, J_n'(x) + (x^2 - n^2) J_n(x) = 0. \tag{58}$$

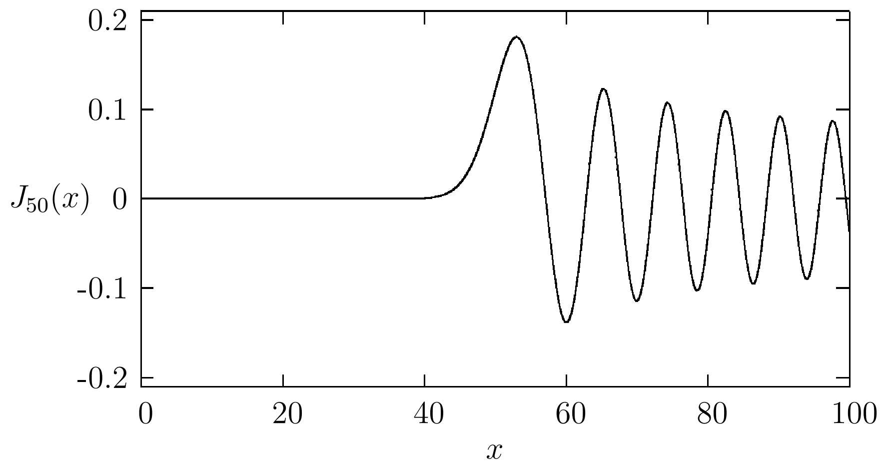


Figure 6: Plot of the Bessel function J_{50}

The asymptotic behavior of J_n is given by the formula

$$J_n(x) = \sqrt{\frac{2}{\pi\, x}} \cos\left(x - \left(n + \frac{1}{2}\right)\frac{\pi}{2}\right) + O\left(\frac{1}{x^{1.5}}\right), \tag{59}$$

indicating that for sufficiently large x any Bessel function oscillates with a frequency of approximately 1.

Reducing equation (58) to the standard form (1) by introducing the auxiliary variable $\alpha = J_n'$ yields a system of two coupled equations. We solve this system for the Bessel function J_{50} in the interval $[50, 15000]$, where the initial values $J_{50}(50)$ and $J_{50}'(50)$ are given. A graph of J_{50} on the interval $[0, 100]$ is depicted in Figure 6.

Tables 5, 6, 7 and 8 summarize the results for the schemes DC1+PC1, DC2+PC2, DC3+PC3 and DC4+PC4, respectively. Each table lists the following measures of performance. Column 1 shows the number of nodes n_s in the equidistant discretization of $[50, 15000]$, while Column 2 shows the corresponding step-size h. Column 3 lists the number of evaluations of the right hand side of (58) required by the deferred correction scheme to compute the k starting values. Furthermore, Column 4 lists the number of evaluations required by the predictor-corrector scheme to compute the $n_s - k$ remaining values. Column 5 shows the total CPU time in seconds, where the timings have been measured on an Intel Pentium 4, 3.2 GHz with 2 GB RAM, as mentioned above. The computations via the schemes DC1+PC1, DC2+PC2 and DC3+PC3 have been performed in double (16-digit) precision, while the computations via the scheme DC4+PC4 have been performed in extended (32-digit) precision.

Finally, Column 6 lists the l^2-error of the resulting solution, which was computed as follows. If φ denotes the analytical solution of the Bessel differential equation, and $\widetilde{\varphi}$ denotes the numerical solution, obtained at the discretization $t_1, \ldots, t_{n_s}$, then we compute the l^2-error of the numerical solution in question at the last 200 nodes as

$$\varepsilon_{l^2}(\widetilde{\varphi}) = \sqrt{\frac{\sum_{k=n_s-200}^{n_s} |\varphi(t_k) - \widetilde{\varphi}(t_k)|^2}{\sum_{k=n_s-200}^{n_s} |\varphi(t_k)|^2}}. \tag{60}$$

n_s	h	N_{DC}	N_{PC}	t/sec	ε_{l^2}
$38,000$	0.3934	345	$75,956$	0.027	1.18×10^{-02}
$42,000$	0.3560	345	$83,956$	0.029	1.67×10^{-03}
$46,000$	0.3250	280	$91,956$	0.031	1.92×10^{-04}
$50,000$	0.2990	280	$99,956$	0.034	2.19×10^{-06}

Table 5: Performance of the schemes DC1+PC1 on the Bessel differential equation

n_s	h	n_{DC}	n_{PC}	t/sec	ε_{l^2}
$52,000$	0.2875	$1,490$	$103,880$	0.078	4.99×10^{-06}
$56,000$	0.2670	$1,311$	$111,880$	0.085	5.39×10^{-08}
$60,000$	0.2492	$1,311$	$119,880$	0.089	4.27×10^{-09}
$64,000$	0.2336	$1,311$	$127,880$	0.096	2.91×10^{-10}
$68,000$	0.2199	$1,132$	$135,880$	0.103	5.40×10^{-11}

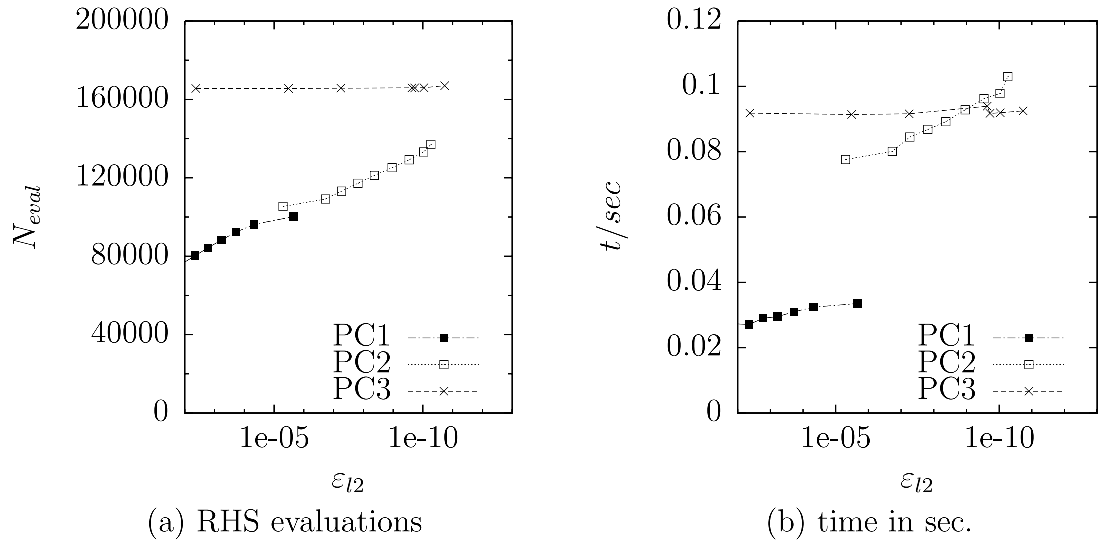
Table 6: Performance of the schemes DC2+PC2 on the Bessel differential equation

n_s	h	n_{DC}	n_{PC}	t/sec	ε_{l^2}
$82,500$	0.1812	665	$164,916$	0.092	3.26×10^{-06}
$82,550$	0.1811	665	$165,016$	0.092	5.68×10^{-08}
$82,600$	0.1810	790	$165,116$	0.092	2.33×10^{-10}
$82,650$	0.1809	665	$165,216$	0.092	1.90×10^{-10}
$82,700$	0.1808	665	$165,316$	0.092	9.27×10^{-11}

Table 7: Performance of the schemes DC3+PC3 on the Bessel differential equation

n_s	h	n_{DC}	n_{PC}	t/sec	ε_{l^2}
$115,000$	0.1300	$2,707$	$229,840$	45.9	2.89×10^{-17}
$120,000$	0.1246	$2,468$	$239,840$	47.9	6.17×10^{-18}
$130,000$	0.1150	$2,468$	$259,840$	51.9	3.27×10^{-19}
$140,000$	0.1068	$2,468$	$279,840$	55.9	1.95×10^{-20}
$150,000$	0.0996	$2,468$	$299,840$	59.8	1.22×10^{-21}
$160,000$	0.0934	$2,229$	$319,840$	63.9	7.48×10^{-23}
$170,000$	0.0879	$2,229$	$339,840$	67.7	3.90×10^{-24}
$180,000$	0.0830	$2,229$	$359,840$	71.7	1.42×10^{-25}
$190,000$	0.0786	$2,229$	$379,840$	75.5	3.63×10^{-27}
$195,000$	0.0766	$2,229$	$389,840$	77.5	3.83×10^{-27}

Table 8: Performance of the schemes DC4+PC4 on the Bessel differential equation



(a) RHS evaluations (b) time in sec.

Figure 7: Performance comparison for the Bessel equation

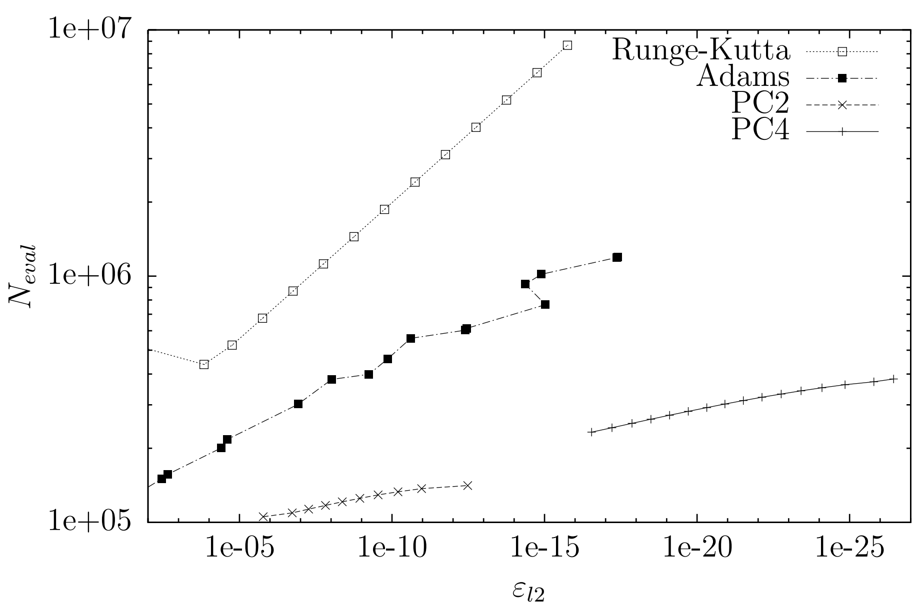


Figure 8: Required RHS evaluations for the Bessel equation

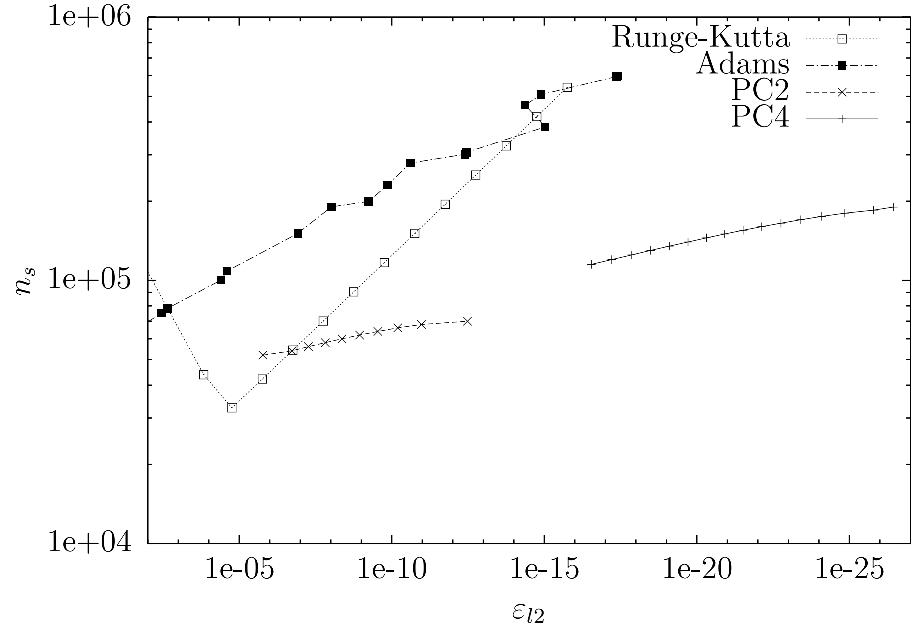


Figure 9: Number of steps for the Bessel equation

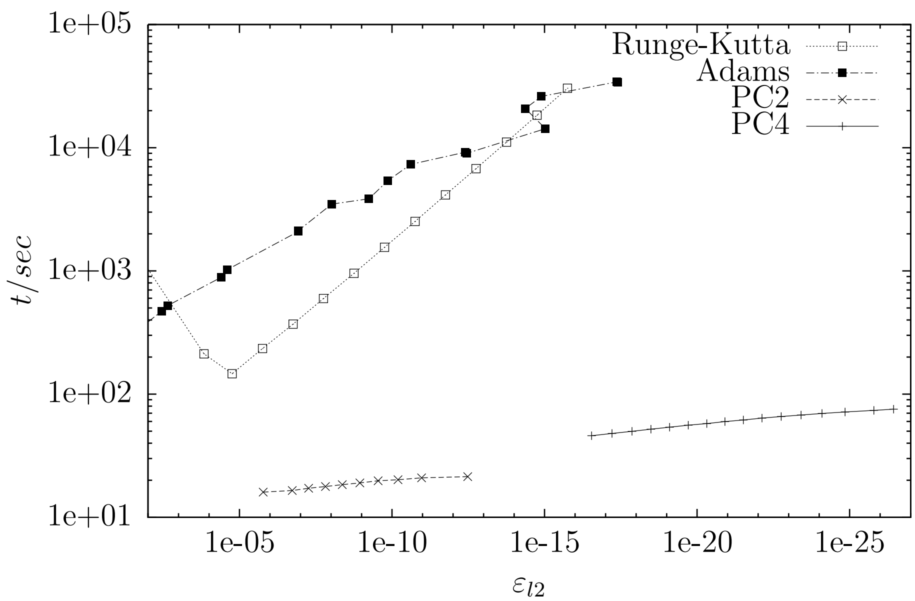


Figure 10: Computation times for the Bessel equation

In our numerical calculations the values of $\varphi = J_{50}$ are computed via a standard numerical procedure for the evaluation of the Bessel function (see, for example, [1]).

Figures 7(a) and 7(b) illustrate the difference in performance of the schemes DC1+PC1, DC2+PC2 and DC3+PC3 for solving the Bessel differential equation (58) of order 50 with double (16-digit) precision. Figure 7(a) shows the total number of evaluations $N_{eval} = n_{DC} + n_{PC}$ of the right hand side of (1) versus the l^2-error ε_{l^2} as defined in equation (60), i.e. the obtained accuracy. Figure 7(b) shows the total CPU time taken by the calculation versus the l^2-error ε_{l^2}.

Finally, Figures 8-10 show a comparison of the performance of the schemes PC3 and PC4 to the performance of the Mathematica implementations of a Runge-Kutta scheme and an Adams-Bashforth-Moulton scheme, as described above. Figure 8 shows the total number of evaluations $N_{eval} = n_{DC} + n_{PC}$ of the right hand side of (1) versus the achieved accuracy (the l^2-error ε_{l^2}). Figure 9 shows the number of steps n_s in the integration interval versus the achieved accuracy, and Figure 10 shows the total CPU time t versus the achieved accuracy. All axes in Figures 8-10 use a logarithmic scale and all the computations for these figures have been performed in extended (32-digit) precision.

Remark 21 The kinks in the graphs for the Runge-Kutta and Adams schemes are artifacts due to the adaptive order and step-size control.

Several observations derived from the above results are summarized at the end of this section.

5.2.2 Jacobi elliptic functions

The Jacobi elliptic functions sn, cn, dn are another commonly used example for a non-stiff problems (see, for example, [1, 10, 6]). The Jacobi elliptic functions with parameter $m \in \mathbb{R}$ satisfy the equations

$$\begin{aligned} sn'(t) &= cn(t) \cdot dn(t) && (61) \\ cn'(t) &= -sn(t) \cdot dn(t) && (62) \\ dn'(t) &= -m \cdot sn(t) \cdot cn(t). && (63) \end{aligned}$$

We apply our schemes to solving this system on the interval $[0, 2000]$ for the case $m = 0.5$ with the initial values $sn(0) = 0$, $cn(0) = 1$, $dn(0) = 1$. The analytical solution for this case can be expressed as the series' (see, for example, [1])

$$sn(t) = \frac{2\pi}{\sqrt{1/2}K} \sum_{n=0}^{\infty} \frac{q^{n+1/2}}{1 - q^{2n+1}} \sin((2n+1)v), \tag{64}$$

$$cn(t) = \frac{2\pi}{\sqrt{1/2}K} \sum_{n=0}^{\infty} \frac{q^{n+1/2}}{1 + q^{2n+1}} \cos((2n+1)v), \tag{65}$$

$$dn(t) = \frac{\pi}{2K} + \frac{2\pi}{K} \sum_{n=1}^{\infty} \frac{q^{n}}{1 + q^{2n}} \cos(2nv), \tag{66}$$

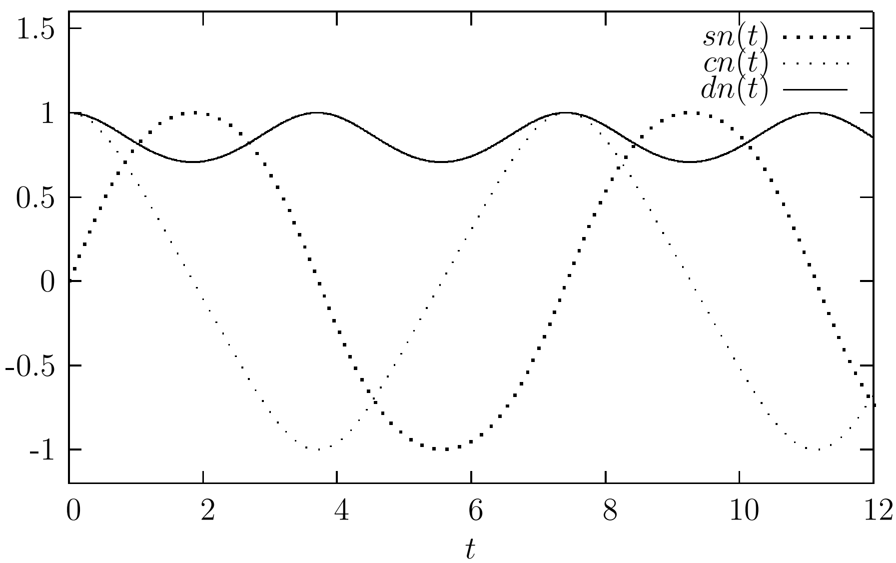


Figure 11: Plot of the Jacobi elliptic functions sn, cn, dn with $m = \frac{1}{2}$

where $q = e^{-\pi}$, $v = \frac{\pi t}{2K}$ and

$$K = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - \frac{1}{2}\sin^2\theta}} \approx 1.85. \tag{67}$$

The first 25 terms of the series' (64), (65), (66) are sufficient to yield the result with 32-digit accuracy. Figure 11 shows a plot of the Jacobi elliptic functions with $m = 0.5$ in the interval $[0, 12]$; they are periodic with period $4\,K$.

Analogous to the previous example, we have used the schemes DC1+PC1, DC2+ PC2, DC3+PC3 and DC4+PC4 listed in Tables 2 and 3 with one corrector step per time step to obtain a solution to system (61), (62), (63) at various equidistant discretization of the interval $[0, 2000]$. The computations for the schemes DC1+PC1, DC2+PC2, DC3+PC3 have been performed in double (16-digit) precision, while the computations for the scheme DC4+PC4 have been performed in extended (32-digit) precision. The results of these tests are summarized in Tables 10-12.

As for the previous example, Column 1 of each table lists the number of nodes n_s in the equidistant discretization of the integration interval, while Column 2 shows the resulting step-size h; Column 3 lists the number of evaluations of the right hand side of the system (61), (62), (63) required by the deferred correction scheme to compute the k required starting values, and Column 4 lists the number of evaluations required by the predictor-corrector scheme to compute the $n_s - k$ remaining values.

Column 5 lists the average l^2-error, which was computed as follows. If φ denotes the analytical solution of any of the functions sn, cn or dn, obtained via series (64), (65) or (66), and $\widetilde{\varphi}$ denotes the corresponding numerical solution, obtained at the discretization $t_1, \ldots, t_{n_s}$, then the l^2-error of this numerical solution is given by equation (60). The values of φ are computed in extended precision (32-digits) using the 25 first terms of the series' (64), (65), (66). In all of our computations the errors $\varepsilon_{l^2}(sn)$, $\varepsilon_{l^2}(cn)$, $\varepsilon_{l^2}(dn)$ were of approximately the same size. In Tables 10-12, Column 5 reports the average

$$\bar{\varepsilon}_{l^2} = \frac{1}{3}(\varepsilon_{l^2}(sn) + \varepsilon_{l^2}(cn) + \varepsilon_{l^2}(dn)). \tag{68}$$

Figure 12 illustrates graphically the difference in the number of required function evaluations of the schemes DC1+PC1, DC2+PC2, DC3+PC3, DC4+PC4 for various accuracies on a logarithmic scale. As for the tables, the results in this figure for the schemes DC1+PC1, DC2+PC2 and DC3+PC3 have been computed in double (16-digit) precision, while the results for the scheme DC4+PC4 have been computed in extended (32-digit) precision. Finally, Figures 13-15 show a peformance comparison between the scheme DC4+PC4 and the Mathematica implementations of a Runge-Kutta scheme and an Adams-Bashforth-Moulton scheme. Figure 13 shows the number of evaluations of the right hand side of system (61), (62), (63) versus the achieved accuracy (the average l^2-error $\bar{\varepsilon}_{l^2}$). Figure 14 shows the number of taken steps n_s versus the achieved accuracy and Figure 15 shows the total CPU time versus the achieved accuracy. All computations for Figures 13-15 have been performed in extended (32-digit) precsion and the axes of Figures 13-15 use logarithmic scales.

5.2.3 Observations

The numerical experiments reported in the preceding two subsections (as well as other numerical experiments we have performed) motivate the following observations.

1. For any of the predictor-corrector schemes (with a fixed number of correction steps) there exists a problem-dependend range of step-sizes for which the scheme behaves similiar to a classically convergent scheme: any decrease in step-size h will increase the accuracy. However, once the step-size h is sufficiently small, further decreasing the step-size will not result in further accuracy improvements. On the other hand, if the step-size becomes too large, the scheme will become unstable. For example, in the Bessel case the scheme PC2 becomes stable at the step-sizes $h < 0.29$ and decreasing the step-size from $h = 0.29$ to $h = 0.22$ will will improve the accuracy from 6 digits to 11 digits. The scheme PC3 becomes stable at $h = 0.18$, where it achieves 10 digits and further decreases in the step-size h will not improve that accuracy. If the scheme PC3 is run with two correction steps, its stability region will increase and it will achieve reasonable accuracies for step-sizes bigger than $h = 0.18$.

2. The performance of the predictor-corrector schemes in the examples above could have been predicted from the accuracy and stability domains shown in Figures 4 and 5. For example, in the Bessel case where the maximal frequency $\lambda \approx 1$ Figure 4(c) shows that the scheme PC2 with 1 correction becomes stable for steplengths $h < 0.3$. Furthermore, Figure 4(d) shows that at $h \approx 0.3$ the scheme will achieve 8 digits and at $h \approx 0.22$ it will achieve 15 digits. The fact that Table 6 shows slightly lower accuracies is due to round-off errors, which accumulated over the long integration interval.

n_s	h	n_{DC}	n_{PC}	t/sec	$\bar{\varepsilon}_{l^2}$
$8,000$	0.2500	670	$15,956$	0.0069	1.59×10^{-01}
$10,000$	0.2000	540	$19,956$	0.0080	2.51×10^{-02}
$12,000$	0.1667	410	$23,956$	0.0094	1.60×10^{-02}
$14,000$	0.1429	345	$27,956$	0.0108	1.88×10^{-03}
$16,000$	0.1250	345	$31,956$	0.0126	3.67×10^{-04}
$18,000$	0.1111	280	$35,956$	0.0140	2.31×10^{-04}

Table 9: Performance of the schemes DC1+PC1 on the Jacobi differential equations

n_s	h	n_{DC}	n_{PC}	t/sec	$\bar{\varepsilon}_{l^2}$
$10,000$	0.2000	$3,638$	$19,880$	0.019	3.90×10^{-01}
$14,000$	0.1429	$2,564$	$27,880$	0.025	3.48×10^{-02}
$18,000$	0.1111	$2,206$	$35,880$	0.033	4.96×10^{-03}
$22,000$	0.0909	$1,848$	$43,880$	0.039	6.51×10^{-04}
$26,000$	0.0769	$1,490$	$51,880$	0.047	3.32×10^{-05}
$30,000$	0.0667	$1,132$	$59,880$	0.053	7.94×10^{-07}
$34,000$	0.0588	953	$67,880$	0.060	2.17×10^{-08}
$38,000$	0.0526	953	$75,880$	0.068	5.96×10^{-10}
$42,000$	0.0476	953	$83,880$	0.075	4.86×10^{-12}
$46,000$	0.0435	953	$91,880$	0.082	2.13×10^{-12}

Table 10: Performance of the schemes DC2+PC2 on the Jacobi differential equations

n_s	h	n_{DC}	n_{PC}	t/sec	$\bar{\varepsilon}_{l^2}$
$16,000$	0.1250	$1,415$	$31,916$	0.021	5.04×10^{-03}
$20,000$	0.1000	$1,040$	$39,916$	0.026	2.39×10^{-06}
$24,000$	0.0833	790	$47,916$	0.032	7.49×10^{-08}
$28,000$	0.0714	665	$55,916$	0.036	6.31×10^{-09}
$32,000$	0.0625	665	$63,916$	0.042	1.49×10^{-09}
$36,000$	0.0556	665	$71,916$	0.047	3.11×10^{-11}
$40,000$	0.0500	665	$79,916$	0.052	1.73×10^{-11}

Table 11: Performance of the schemes DC3+PC3 on the Jacobi differential equations

n_s	h	n_{DC}	n_{PC}	t/sec	$\bar{\varepsilon}_{l^2}$
$20,000$	0.1000	6292	$39,840$	12.1	1.44×10^{-02}
$40,000$	0.0500	3424	$79,840$	23.7	7.02×10^{-11}
$60,000$	0.0333	2229	$119,840$	35.4	1.07×10^{-15}
$80,000$	0.0250	2229	$159,840$	47.1	5.37×10^{-19}
$100,000$	0.0200	1990	$199,840$	58.8	6.76×10^{-22}
$120,000$	0.0167	1990	$239,840$	70.5	3.23×10^{-24}
$140,000$	0.0143	1990	$279,840$	82.3	9.05×10^{-26}
$160,000$	0.0125	1751	$319,840$	94.2	5.15×10^{-27}
$180,000$	0.0111	1751	$359,840$	106	5.73×10^{-27}

Table 12: Performance of the schemes DC4+PC4 on the Jacobi differential equations

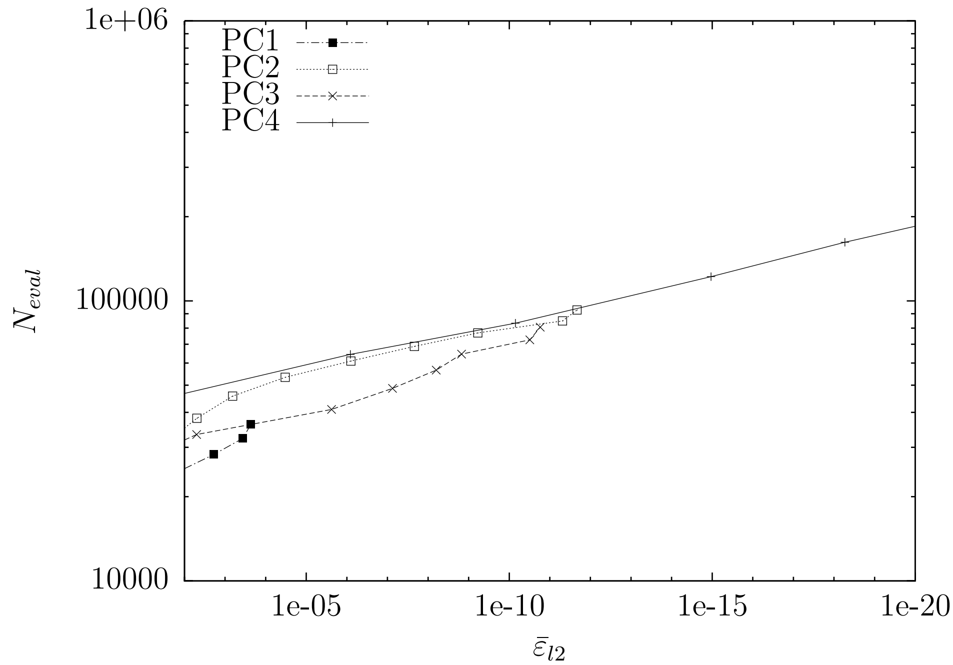


Figure 12: Required RHS evaluations for the Jacobi elliptic functions

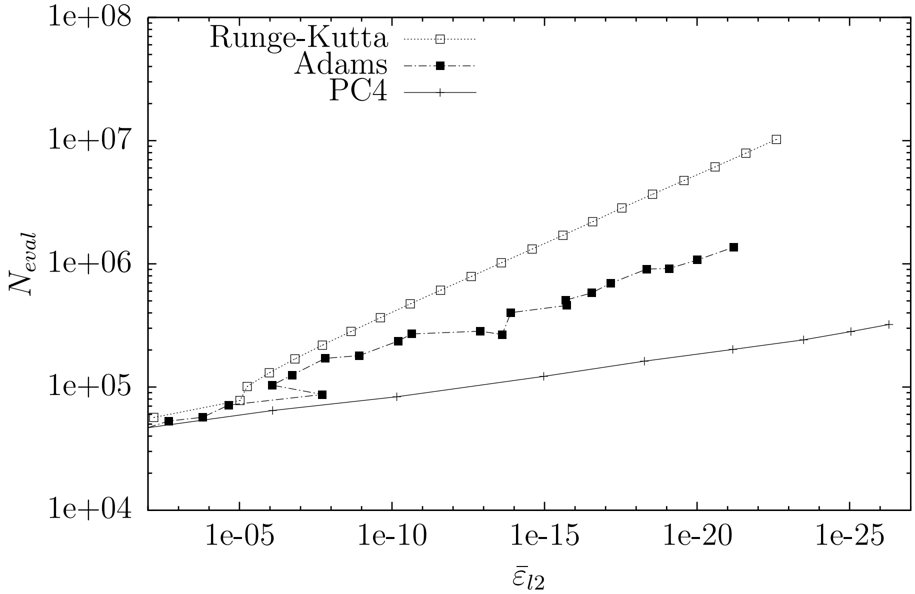


Figure 13: Required RHS evaluations for the Jacobi elliptic functions

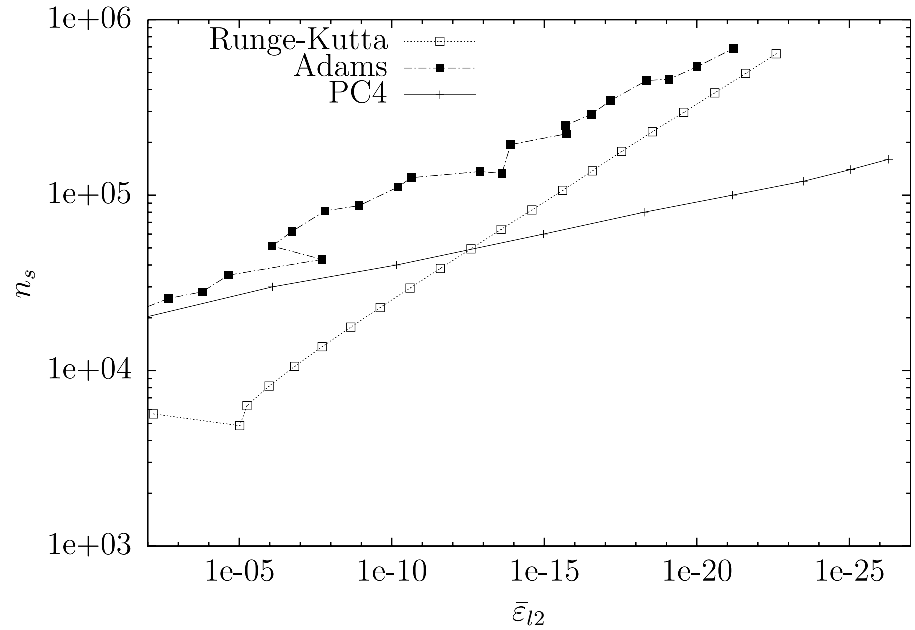


Figure 14: Number of steps for the Jacobi elliptic functions

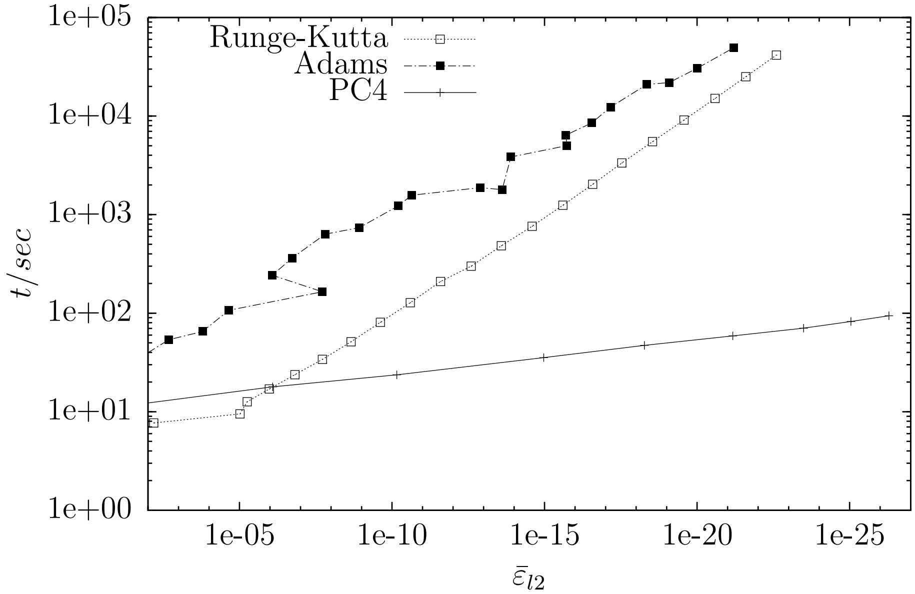


Figure 15: Computation times for the Jacobi elliptic functions

3. The convergence rate of the predictor-corrector schemes of this paper is signifincantly higher than the convergence rate of a high-order Runge-Kutta method or a high-order Adams predictor corrector method (See Figures 8-10 and 13-15, please note that the scales are logarithmic). In terms of the number of steps the Runge-Kutta scheme does best for low accuracies; for higher accuracies (12 digits or more) the schemes of this paper become superior. In terms of the number of RHS evaluations and the actual timings our schemes are superior as soon as 5 digits or more are required.

6 Conclusions and future work

We have presented a new class of predictor-corrector schemes, along with spectral deferred correction schemes for their initialization. Our experiments indicate that on a wide range of problems the schemes of this paper outperform serveral state-of-the-art ODE solvers, particularly when high accuracy is required.

This paper discusses explicit schemes, which are applicable to non-stiff problems. Extension of this work to stiff environments is in progress, and will be reported at a later date.

References

[1] M. Abramovitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover Publications, Mineola, New York, 1972.

[2] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, 1998.

[3] P. Brock and F. J. Murray, *The use of exponential sums in step by step integration*, Mathematical Tables and Other Aids to Computation, 6 (1952), pp. 63–78.

[4] H. Cheng, Z. Gimbutas, P. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM Journal on Scientific Computation, 26 (2005), pp. 1389–1404.

[5] G. Dahlquist and Å. Björk, *Numerical Methods*, Dover Publications, Mineola, New York, Dover ed., 2003.

[6] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, Journal BIT Numerical Mathematics, 40 (2000), pp. 241–266.

[7] W. Gautschi, *Numerical integration of ordinary differential equations based on trigonometric polynomials*, Numerische Mathematik, 3 (1961), pp. 381–397.

[8] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, third ed., 1996.

[9] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing qr factorization*, SIAM Journal on Scientific Computation, 17 (1996), pp. 848–869.

[10] E. Hairer and G. Wanner, *Solving ordinary differential equations I, Non-stiff Problems*, Springer, 1993.

[11] ——, *Solving ordinary differential equations II, Stiff and differential-algebraic problems*, Springer, 1996.

[12] A. C. Hansen and J. Strain, *Convergence theory for spectral deferred correction*, Preprint, (2006).

[13] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, Journal of Computational Physics, 214 (2006), pp. 633–656.

[14] A. Iserles, *A first course in the numerical analysis of differential equations*, Camebridge University Press, 1996.

[15] W. H. Press, S. A. Teulkolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77*, Press Syndicate of the University of Cambridge, 2001, ch. 4.5.