A Highly Accurate Solver for Stiff Ordinary Differential Equations

Dan Kushnir* Vladimir Rokhlin†

September 1, 2011

Abstract

We introduce a solver for stiff ordinary differential equations (ODEs) that is based on the deferred correction scheme for the corresponding Picard integral equation. Our solver relies on the assumption that the solution can be accurately represented by a combination of carefully selected complex exponentials. The solver's accuracy and stability rely on the computation of highly accurate quadrature weights for the integration of the selected exponentials on equidistant nodes. We analyze our solver stability and accuracy regions, and demonstrate its fast convergence on stiff problems. The solver is combined with an adaptive step-size scheme employing interpolation formulas for the exponentially fitted solution.

1 Introduction

The development of numerical solvers for ordinary differential equations (ODEs) is a wide and mature area of research. In particular for non-stiff ODEs, many successful approaches have been studied and used, which provide excellent efficiency [11]. For stiff ODEs a more careful treatment is required, however, many successful methods have been suggested [12]. Direct, high order solvers such as Runge-Kutta and Adams methods [11] have been widely used. They are based on the approximation of the solution by a polynomial, an approach that is too expensive when high accuracy is required. A different class of solvers is based on the iterative use of low-order methods to improve a solution approximation, of which our particular interest is in the deferred correction method [2], [3], [4]. However, due to lack of stability the use of the classical deferred corrections method has been somewhat limited. The spectral deferred correction scheme suggested in [1] offers a significant improvement in stability and accuracy over the classical one. The improvement is obtained by considering the iterative spectral solution of the corresponding Picard integral equation on a Gauss-Legendre grid. The solution of the Picard equation eliminates the need for numerical differentiation which is unstable. Following the work of [1] came other works employing different quadrature rules and modifying the scheme to a semi-implicit one [6], [7], further acceleration of the method has also been suggested in [8], [9] and [10].

*Dept. of Mathematics, Yale University, New Haven CT 06511
†Dept. of Mathematics, Yale University, New Haven CT 06511

In this paper we describe an accurate deferred correction based solver for the Picard equation corresponding to stiff ODEs in the initial value problem

$$\varphi'(t) = F(t, \varphi(t)), \quad \varphi(a) = \varphi_a \quad t \in [a, b]. \tag{1}$$

The solution is approximated on a discretized grid of *uniform size time steps*. Our key assumption for the solution of (1) is that the solution can be accurately approximated by a combination of complex exponentials. Such an approach, known as exponentially fitted methods, is different than the typical polynomial fitted methods. Exponentially fitted methods have been used for the solution of differential equations in various works (e.g. [13], [14], and [15]). Similarly to the non-stiff ODEs solver of [16], we use complex exponentials λ_j for the solution approximation

$$\varphi(t) \sim \sum_{j=1}^{n} \alpha_j e^{\lambda_j t}, \tag{2}$$

where the coefficients α_j correspond to the solution being approximated. The choice of $\lambda_j,\ j = 1, ..., n$ is based on a numerical scheme known as skeletonization (see [18]). Once the exponentials are chosen, the coefficients α_j are computed via a least-squares procedure. Our least squares procedure guarantees the accuracy of the resulting quadratures on regions in the complex plane by constructing the quadratures on the boundaries of the regions and using the maximum principle (lemma 2.5 below) to show that their accuracy holds for the whole region.

Our solver is adapted for the stiff case by using implicit deferred corrections schemes. It demonstrates accuracy regions at least as large as the designed quadrature accuracy. In particular, it has an arbitrary order of accuracy which is bounded below by the accuracy of the quadrature rules or by the number of iterations performed. Obviously, the quadratures accuracy can be pre-defined to machine precision. The solver has satisfactory stability regions due to its high accuracy. Specifically, a method of order 10 is A-stable (see section 2.1) and $A(\alpha)$-stable with α close to 90° for a method of order 14.

We compare our method with state of the art deferred correction methods ([6], [7]) where we show an improvement in the algorithm efficiency over methods with different quadrature rules. A comparison with additive Runge-Kutta method [5] is also demonstrated, where our method shows preferable convergence. Finally, we demonstrate the use of an efficient adaptive step-size scheme which initializes finer grids by highly accurate least squares interpolation, designed (as well) to fit the exponential representation of the solution. The adaptive schemes is critical for highly stiff problems and in particular where layers exist.

The paper is organized as follows. In section 2, useful mathematical and numerical preliminaries that are used throughout the paper are briefly described. In section 3 we discuss the mathematical apparatus used to construct the algorithm. In section 4 we describe the algorithm and its adaptive scheme. Section 5 analyzes the accuracy and stability properties of the solver. Section 6 demonstrates the algorithm performance with a few numerical experiments with comparison to related work.

2 Mathematical and Numerical Preliminaries

In this section we describe several facts from numerical analysis to be used throughout the rest of this paper. The set $\{a = t_1, ..., t_n = b\}$ such that $t_i = a + (i-1)h$ will be called an equidistant discretization of $[a, b]$ with step size h. The function $\varphi : \mathbb{R} \to \mathbb{C}^m$ and its derivative $F : \mathbb{R} \times \mathbb{C}^m \to \mathbb{C}^m$ are sufficiently smooth. The complex modulus or absolute value is denoted by $|\cdot|$, and $\|\cdot\|_2$ denotes the Euclidean norm of a vector or the corresponding operator norm of a matrix.

2.1 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 ([11], [12]). Let $\tilde{\varphi}$ denote the solution to the IVP (1), obtained by a numerical solver at equidistant discretization $t_1, ..., t_n$ 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 (1) there exists a constant $M > 0$ such that

$$|\varphi(b) - \tilde{\varphi}(b)| < Mh^p. \tag{3}$$

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

$$\varphi'(t) = \lambda\varphi(t), \quad t \in [0, 1], \tag{4}$$

$$\varphi(0) = 1, \tag{5}$$

where $\lambda \in \mathbb{C}$. The amplification factor $\mathrm{Am}(\lambda)$ for $\lambda \in \mathbb{C}$ is defined by the formula

$$\mathrm{Am}(\lambda) = \tilde{\varphi}(1). \tag{6}$$

If, for given λ, $|\mathrm{Am}(\lambda) \leq 1|$, then the numerical method is said to be stable for that value of λ. The set of values of λ for which the method is stable is called the *stability domain*. If a method is stable for all λ in the left-half plane ($\mathrm{Re}(\lambda) \leq 0$) then the method is said to be *A-stable*. A method is said to be *$A(\alpha)$-stable* if it is stable for all λ such that $\pi - \alpha \leq \arg(\lambda) \leq \pi + \alpha$. Thus, A-stability is equivalent to *$A(\alpha)$-stability* with $\alpha = 90°$. Finally, a method is said to be *L-stable* if it is A-stable and

$$\lim_{\mathrm{Re}(\lambda)\to-\infty} \mathrm{Am}(\lambda) = 0. \tag{7}$$

Remark 2.1. *Increasing the number of discretization steps for a given interval will increase the frequency domain to which the numerical method applies. In general, changing the step size of a numerical scheme for the solution of (1) from 1 to h is equivalent to changing the exponent λ in (1) to λh, as the dilation $\tau = th$ transforms (4) into*

$$\frac{\partial\varphi(\tau)}{\partial\tau} = \lambda h\varphi(\tau). \tag{8}$$

Remark 2.2. *If the accuracy domain of a numerical scheme for a prescribed precision ε contains the semi-disk*

$$S = \{\lambda \in \mathbb{C} : |\lambda| \leq |\tilde{\lambda}|\}, \tag{9}$$

then the performance of the scheme is measured by the number of steps per wavelength

$$\frac{2\pi}{|\tilde{\lambda}|}. \tag{10}$$

2.2 SVD and Least Squares

Given the linear system

$$Ax = b, \tag{11}$$

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

$$||Ax - b||_2 \tag{12}$$

is called a least squares solution of the system (11). If X is the set of least squares solutions of (11) then X contains one unique element of minimum L_2-norm

$$x_{LM} = \text{argmin}\{||x||_2 : x \in X\}, \tag{13}$$

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. $U \in C^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are orthonormal and $\Sigma \in \mathbb{R}^{n\times n}$ is diagonal, then

$$x_{LM} = \sum_{i=1}^{r} \frac{u_i^* b}{\sigma_i} v_i, \tag{14}$$

where $(u_1, ..., u_m)$ are the columns of U, $(v_1, ..., v_n)$ are the columns of V, $(\sigma_1, ..., \sigma_r)$ 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 $(\sigma_1, ..., \sigma_{min(m,n)})$ 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 (11) as

$$x_{LM}(\varepsilon) = \sum_{i=1}^{r(\varepsilon)} \frac{u_i^* b}{\sigma_i} v_i, \tag{15}$$

where $r(\varepsilon)$ is the numerical rank of precision ε of A and all other quantities are as in equation (14). Given an algorithm for the computation of the SVD, the minimum norm least squares solution $x_{LM}(\varepsilon)$ can be computed via (15). 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 [17].

2.3 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 [18] in slightly more general form.

Lemma 2.3. *Given a matrix $A \in \mathbb{C}^{m\times n}$ with columns $(a_1, ..., a_n)$, and $k \in \mathbb{N}$ such that $1 \leq k < min(m, n)$, there exists a selection of k columns of A such that*

$$A = (a_{n_1}, ..., a_{n_k})T + X, \tag{16}$$

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

$$\|X\|_2 \leq \sigma_{k+1}(A)\sqrt{1 + (k(min(m, n) - k))}. \tag{17}$$

In other words, the lemma states that A can be approximated by interpolating between k of A's columns, if the $(k+1)$st 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 2.4. *The factorization (16) 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 [18].*

2.4 Implicit Euler Method

The 1st order Euler methods solve the initial value problem (1) on the interval $[t_1, t_1 + L]$ by taking a sequence of $n-1$ steps $t_2, ..., t_n$ such that $t_{i+1} = t_i + h$, where $h = L/(n-1)$. In this paper the focus is on stiff ordinary differential equations that can be solved by implicit methods only. Specifically, we describe for this matter the implicit Euler scheme which uses the backward difference quotient

$$y_i = y_{i-1} + h \cdot F(t_i, y(t_i)). \tag{18}$$

The scheme (18) requires the computation of the derivative at t_i and solving a system of equations for y_i. The solution requires the inversion of a matrix constructed from the corresponding Jacobian(s), e.g. by a direct method.

2.5 The Bisection Method

The bisection method finds the root of the 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 the midpoint

$$x_3 = \frac{x_1 + x_2}{2}. \tag{19}$$

If $sign(f(x_3)) = sign(f(x_2))$ we set $x_2 = x_3$, otherwise $x_1 = x_3$, and the search is repeated. The method stops when x_1 and x_2 are sufficiently close. To obtain a root with precision ε the method requires $O(log(\varepsilon))$ steps.

2.6 Newton's Method

Consider a general system of equations in n unknowns:

$$f_i(x_1, x_2, ..., x_n) = 0, \;\; i = 1, 2, ..., n. \tag{20}$$

Newton's method for solving (20) is derived from Taylor's formula:

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + O(|x - x_k|^2), \tag{21}$$

in which the quadratic term is neglected. One can then use the iteration

$$(x_{k+1} - x_k)f'(x_k) + f(x_k) = 0, \quad k = 0, 1, 2, ... \tag{22}$$

to solve a linear system of equations for x_{k+1}. The method requires the inversion of Jacobian matrix $f'(x_k)$ which can be computed by direct methods [19].

2.7 The Maximum Modulus Principle

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

Lemma 2.5. *(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.*

$$\max_{\partial D} |f| = \max_{D} |f|. \tag{23}$$

3 Analytical Apparatus

In this section we describe the Picard Integral Equation, and a numerical method to solve it. We then present an accurate method used within the suggested solution to the Picard Equation, to represent and approximate band limited functions at equidistant nodes.

3.1 The Picard Integral Equation

The spectral deferred correction schemes presented in [1] relay on the integral form of the solution of (1). Integrating equation (1) with respect to t yields the Picard integral equation

$$\varphi(t) = \varphi_a + \int_a^t F(\tau, \varphi(\tau))d\tau. \tag{24}$$

Let $\varphi^0(t)$ be some approximate solution to (1). A measure for the quality of the approximation is given by the residual function $\varepsilon(t)$:

$$\varepsilon(t) = \varphi_a + \int_a^t F(\tau, \varphi^0(\tau))d\tau - \varphi^0(t). \tag{25}$$

We define the error $\delta(t)$ by

$$\delta(t) = \varphi(t) - \varphi^0(t). \tag{26}$$

Substituting (26) into (24) we obtain

$$\varphi^0(t) + \delta(t) = \varphi_a + \int_a^t F(\tau, \varphi^0(\tau) + \delta(\tau))d\tau. \tag{27}$$

Substituting (25) into (27) yields a *correction equation*

$$\delta(t) = \int_a^t [F(\tau, \varphi^0(\tau) + \delta(\tau)) - F(\tau, \varphi^0(\tau))]d\tau + \varepsilon(t). \tag{28}$$

With the function $G : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ defined as

$$G(t, \delta) = F(t, \varphi^0(t) + \delta(t)) - F(t, \varphi^0(t)), \tag{29}$$

we can rewrite (28) as a Picard-type integral equation

$$\delta(t) - \int_a^t G(\tau, \delta(\tau))d\tau = \varepsilon(t). \tag{30}$$

The Picard-type error equation is also the basis for spectral deferred correction methods as, e.g., in [1].

3.2 Euler Methods for the Picard Equation

Let $a = t_1, ..., t_n = b$ be a discretization of the interval $[a, b]$. Similar to the construction of the Euler method for the solution of (1) (as presented in section 2.4), the backward Euler solution for (30) is given by

$$\delta_i = \delta_{i-1} + h_{i-1} \cdot G(t_i, \delta_i) + (\varepsilon(t_i) - \varepsilon(t_{i-1})). \tag{31}$$

Each step for computing δ_i is referred to in the following as the *Picard Integral Exponential Solver* (*PIES*) step. Similarly to the implicit Euler method, the solution for the correction δ_i requires an inversion of a matrix for the corresponding system of equations, typically done by a direct method. A marching scheme as (31) for the numerical computation of a solution for (1) will require a high-order accurate method for the integration involved in computing the residual ε in (25). An accurate integration method is described in the following sections 3.3, 3.4.

3.3 Highly Accurate Quadrature Rules

In this section we describe the computation of highly accurate quadrature weights for the approximation

$$\int_{-1}^{t_j} \varphi(\tau)d\tau \approx \sum_{i=1}^{k} \varphi(t_i)\omega_{ij}, \tag{32}$$

where $\omega_{ij} \in \mathbb{R}$, and $t_1, ..., t_k$ is a discretization of $[-1, 1]$. Our computation is based on the assumption that the ODE solution φ is a linear combination of the finite set of exponentials $e^{\lambda_1 t}, e^{\lambda_2 t}, ..., e^{\lambda_n t}$. Our process for choosing the complex $\lambda_1, \lambda_2, ..., \lambda_n$ is described in section 3.4 below. The matrix $\omega = (\omega_{ij})$ has to satisfy the linear system

$$A\omega = b, \tag{33}$$

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

$$A = \begin{pmatrix} e^{\lambda_1 t_1} & e^{\lambda_1 t_2} & \dots & e^{\lambda_1 t_k} \\ e^{\lambda_2 t_1} & e^{\lambda_2 t_2} & \dots & e^{\lambda_2 t_k} \\ \vdots & \vdots & & \vdots \\ e^{\lambda_n t_1} & e^{\lambda_n t_2} & \dots & e^{\lambda_n t_k} \end{pmatrix} \tag{34}$$

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

$$b = \begin{pmatrix} \int_{-1}^{t_1} e^{\lambda_1 \tau} d\tau & \int_{-1}^{t_2} e^{\lambda_1 \tau} d\tau & \dots & \int_{-1}^{t_k} e^{\lambda_1 \tau} d\tau \\ \int_{-1}^{t_1} e^{\lambda_2 \tau} d\tau & \int_{-1}^{t_2} e^{\lambda_2 \tau} d\tau & \dots & \int_{-1}^{t_k} e^{\lambda_2 \tau} d\tau \\ \vdots & \vdots & & \vdots \\ \int_{-1}^{t_1} e^{\lambda_n \tau} d\tau & \int_{-1}^{t_2} e^{\lambda_n \tau} d\tau & \dots & \int_{-1}^{t_k} e^{\lambda_n \tau} d\tau \end{pmatrix}. \tag{35}$$

Each column of ω is approximated by the minimum norm least squares solution (15) of (33), such that the accuracy of the quadratures is of order ε.

Different types of quadrature rules can be formulated based on the the above scheme which include or exclude the leftmost endpoint. In [7] an L-stable quadrature rule is obtained by omitting the left most node in the interval. In other words $\omega_{1,\cdot} = 0$ in (32). Such a scheme will be referred to as a *right-hand rule* (rhr), whereas including the left hand point will be referred to as a *left-hand rule* (lhr).

3.4 Skeletonization of the Complex Semi-disc

We assume that our solution is well approximated by

$$\varphi(t) \sim \sum_{j=1}^{n} \alpha_j e^{\lambda_j t} \tag{36}$$

for some $\lambda_1, ..., \lambda_n$. The first step in constructing an accurate quadrature rule is to select $\lambda_1, ..., \lambda_n$ in (34) such that (32) will be approximated for all functions $f_\lambda(t) = e^{\lambda t}$ for which λ lies within the complex semi-disk

$$S_\rho = \{\lambda \in \mathbb{C} \mid \mathrm{Re}(\lambda) \leq 0,\ |\lambda| \leq \rho\}. \tag{37}$$

In the following we justify the choice of $\lambda_1, ..., \lambda_n$ from the boundary of S_ρ (denoted by ∂S_ρ).

Lemma 3.1. *Given an initial value problem*

$$\varphi' = \lambda\varphi, \quad \varphi(0) = \varphi_0, \tag{38}$$

let φ_λ denote its solution, and $\tilde{\varphi}_\lambda$ its approximate solution computed via the scheme (31), with quadratures for the residual as described in section 3.1. 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

$$\|\varphi_\lambda(t) - \tilde{\varphi}_\lambda(t)\| < \varepsilon \tag{39}$$

holds for all $\lambda \in \partial D$, then (39) also holds for all $\lambda \in D$.

***Proof**: For any fixed value of t the method suggested in sections 3.1 and 3.2 computes $\tilde{\varphi}_\lambda$ by a finite number of additions and multiplications of linear functions of λ. The function $g_t : \mathbb{C} \to \mathbb{C}$ defined by $\lambda \mapsto \tilde{\varphi}_\lambda$ 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 $\lambda \mapsto \varphi_\lambda$ is also an analytic function of λ, the error term in $\varphi_\lambda - \tilde{\varphi}_\lambda$ is also an analytic function of λ. Thus the Lemma follows from the maximum modulus principle (Lemma 2.5).*$\square$

Since one $PIES$ step depends linearly on the solution at the previous nodes, any method valid for any $e^{\lambda_1 t}, e^{\lambda_2 t}, ..., e^{\lambda_n t}$ is thus valid for any linear combination of these functions in the following sense: Let $\lambda \in \partial S_\rho$ for which there exist coefficients $c_1, ..., c_n \in \mathbb{C}$ such that

$$|e^{\lambda t} - \sum_{i=1}^{n} c_i e^{\lambda_i t}| < \delta, \tag{40}$$

for all $t \in [-1, 1]$. If (39) is satisfied for $\lambda = \lambda_1, \lambda_2 ..., \lambda_n$, then by the triangle inequality

$$|\varphi_\lambda - \tilde{\varphi}_\lambda| < \varepsilon + \delta. \tag{41}$$

Inequality (41) is satisfied for sufficiently dense choice of $\{e^{\lambda t} | \lambda \in \partial S_\rho\}$. Uniform discretization of ∂S_ρ is employed for that matter. In particular, we use the skeletonization process described in Section 3.4 to significantly reduce the number of required nodes (and thus the complexity of solving (11)). Let $\lambda_1, ..., \lambda_N$ and $t_1, ..., t_M$ be uniform dense discretization of ∂S_ρ and $[-1, 1]$, respectively, and the factorization of (34) with error term $||X||_2 < \delta$ yields the k columns with indices $\{i_1, ..., i_k\}$. Then the exponentials $e^{\lambda_{i_1} t}, ..., e^{\lambda_{i_k} t}$ satisfy (41) to precision δ.

Remark 3.2. *Since the integral and its approximation in (32) are analytical functions of λ, it follows that equation (32) also holds for all linear combinations of $\{e^{\lambda t} | \lambda \in S_\rho\}$.*

Remark 3.3. *The skeletonization of (34) corresponds to a step size $h = 2/(k-1)$ for functions of the form $f_\lambda = e^{\lambda t}$ where $\lambda \in S_\rho$. A new discretization of $[-1, 1]$ with step size $\tilde{h} = \alpha h$ will be valid for all solutions of the form $f_\lambda = e^{\lambda t}$ where $\lambda \in S_{\rho/\alpha}$. Thus, our PIES can be applicable to functions of higher frequency by reducing the step size. For example, halving the step size will double the frequency of the functions that can be approximated by the skeletonization of (34).*

Remark 3.4. *ρ determines the maximal frequency of the functions to which the PIES scheme can be applied. Thus, by construction, the number of steps per wavelength to achieve the optimal accuracy of the resulting scheme is approximately $\frac{k\pi}{\rho}$.*

3.5 Interpolation

We construct a highly accurate interpolation scheme for the desired solution of (1) with a discretization $t_1, ..., t_k$

$$\varphi(t_j) = \sum_{i=1}^{k} \varphi(t_i) w_{ij} \quad j = 1, ..., K \;\; k \leq K \tag{42}$$

by solving

$$A\omega = b. \tag{43}$$

for ω via the least squares solution (14) for the matrix $\omega = \omega_{ij}$, where A is defined as in (34), and

$$b = \begin{pmatrix} e^{\lambda_1 t_1} & e^{\lambda_1 t_2} & \dots & e^{\lambda_1 t_K} \\ e^{\lambda_2 t_1} & e^{\lambda_2 t_2} & \dots & e^{\lambda_2 t_K} \\ \vdots & \vdots & & \vdots \\ e^{\lambda_n t_1} & e^{\lambda_n t_2} & \dots & e^{\lambda_n t_K} \end{pmatrix}. \tag{44}$$

4 The Algorithm

In this section we describe the detailed algorithm for solving the IVP (1) by numerically solving the corresponding Picard integral equation. We also describe an efficient scheme employing the Picard solver which allows to adapt the step size to a prescribed error tolerance.

4.1 A Deferred Corrections Solver for the Picard Integral Equation

Our strategy for solving the Picard integral equation (24) is to compute a provisional solution by using the implicit Euler method (see section 2.4), then to iteratively solve the correction equation (28). The input for the $PIES$ algorithm is the provisional solution and the quadrature weights necessary for computing the residual (25) via the scheme described in sections 3.3, 3.4. Each deferred correction iteration consists of two steps: computing the residual function (25), and then computing the corrections for the current solution approximation via the scheme (31). The last step includes Newton iterations (see section 2.6) for the solution of the correction equations (31). The correction is then added to the current approximation and the process is repeated until a prescribed error threshold is met.

Picard Integral Exponential Solver (PIES): The input for the solver are the initial approximation $\varphi^{[0]}$, the quadrature weights ω_{ij}, and the stopping criterion ε_I.

1. $j = 0$
2. Repeat J times or until $|\delta_i^{[j]}| \leq \varepsilon_I$, for $i = 1, ..., n$:
   (a) Compute the residual equation (25) for $i = 1, ..., n$
   (b) Compute the corrections $\delta_i^{[j]}$ via the implicit Euler (31) for $i = 1, ..., n$.
   (c) Update: $\varphi^{[j+1]} = \varphi^{[j]} + \delta^{[j]}$
   (d) $j = j + 1$
3. Return $\varphi^{[j]}$

Definition 4.1. *The solver described above involving J iterations and n nodes is denoted throughout the rest of the paper by $PIES_n^J$. The corresponding solution generated by the scheme is denoted by $PIES_n^J(F, \varphi_a)$.*

Remark 4.2. *$PIES$ is not a convergent numerical scheme. Namely, after a prescribed accuracy is obtained, the scheme will not converge as the step-size is reduced. In practice, since machine precision is finite, the user can always aim at that precision and obtain it regardless of the scheme being non-convergent.*

The usual estimate of the order of accuracy obtained with classical iterated deferred corrections is [4] $O(h^{min((J+1)k,n)})$, where k is the order of the method used to obtain $\varphi^{[0]}$ and solve the correction equation (27). Combining the results from [4] and remark 4.2, it is straightforward to obtain the following result:

Theorem 4.3. *The solver $PIES_n^J$ converges to the exact solution with order of accuracy* $\min(J + 1, \lfloor \log(1/\varepsilon) \rfloor)$ *where ε is the prescribed accuracy of the quadrature weights as defined in section 3.3.*

4.2 Adaptive Implementation

Accuracy and step-size control are an important component in any numerical ODE solver. We implemented an adaptive scheme to reduce the computational work for obtaining a prescribed accuracy. The accuracy control relays on criteria we have used in practice to estimate the

actual error. Moreover, a high order interpolation is used to reduce the number of deferred correction iterations used each time the step size is reduced.

Accuracy control: Since the actual error $|\varphi - PIES_n^J(F, \varphi_a)|$ can not be measured, we describe below several criteria used for accuracy control in our adaptive scheme. In practice the criteria have shown to be very reliable for monitoring the true error of the solver.

1. Internal consistency: we check that the correction process is indeed converging. That is, the magnitude of the corrections δ (31) satisfy $\delta < \varepsilon_I$ at the last correction.
2. Overflow: in some cases the solution is so under-resolved that an overflow is detected. In most applications we set an arbitrary threshold (typically 10^{10}) to verify that such instability in our process does not occur at any time step for every iteration.
3. Step-size variation: we check that the difference at $t = b$ between $PIES_n^J(F, \varphi_a)$ computed with the step size h and $PIES_n^J(F, \varphi_a)$ computed with step size $\bar{h} < h$ are less then ε_{adpt} at the last iteration.

Adaptive deferred corrections: In most cases the interval $[a, b]$ is divided into subintervals $[t_m, t_{m+1}]$. Specifically, the algorithm $PIES_n^J$ is applied on $[t_m, t_{m+1}]$ with the discretization $t_m = t_1, ..., t_n = t_{m+1}$. The approximation obtained at t_{m+1} is then fed as an initial condition for $PIES_n^J(F, \varphi(t_{m+1}))$ applied on the next subinterval $[t_{m+1}, t_{m+2}]$, and so on.

The input to adaptive implementation algorithm are the pre-computed quadratures and the accuracy parameter ε_{adpt}. Then we apply $PIES$ first with step size h and then again with $\bar{h} = h/2$. If the prescribed accuracy ε_{adpt} is met (criterion 3 above), and criterion 1 holds for the last iteration J, then the finer approximation is kept. Otherwise, the step size is halved again and the same procedure repeats. If the accuracy criteria are satisfied for s subsequent subintervals (s is typically 2), then the step size is doubled. The finer iterations are initialized with the interpolated values from the coarser approximation. At any stage if an interpolation from coarse approximation is not possible a provisional solution is computed by backward Euler method. The interpolation is computed as described in section 3.5. The adaptive algorithm is described below.

a. ***for*** $t_m = a, m = 1, 2, ...$
   1. ***if*** $t_m = b$ return φ
   2. Initialize $\varphi^{[0]}$
   3. Compute $PIES_n^J(F, \varphi_{t_m}^{[0]})$ for step size h with discretization $t_m = t_1, ..., t_n = t_{m+1}$ (monitor overflow)
   4. Initialize $\varphi^{[0]}$ for step size $\bar{h} = h/2$ by interpolation (see section 3.5)
   5. Compute $PIES_n^J(F, \varphi_{t_m}^{[0]})$ for step size $\bar{h}$
   6. ***if*** both approximations agree to accuracy ε_{adpt} and $\delta^{[J]} < \varepsilon_I$
      i. ***if*** $s' = s:\ h = 2h,\ t_m = t_{m+1},$
      ii. ***if*** $s' < s:\ t_m = t_{m+1},\ s' = s' + 1$
   7. ***else*** $s' = 0,\ h = \bar{h}.$

b. return $\varphi^{[J]}$

5 Stability and Accuracy Analysis

In this section we investigate the stability and accuracy properties of different $PIES$ schemes implemented. The chosen schemes are elaborated in Table 1.

Scheme	ρ	n	$\varepsilon+\delta$	J
DCL1	3.15	10	10^{-5}	4
DCL2	3.15	34	10^{-15}	13
DCL2B	3.15	34	10^{-15}	9
DCL3	3.15	52	10^{-24}	22
DCL4	6.30	42	10^{-15}	13
DCR2B	3.15	34	10^{-15}	9

Table 1: deferred corrections schemes.

DCL schemes correspond to schemes where a left hand quadrature rule is used (see section 3.3), while in DCR schemes a right hand quadrature rule is used. n denotes the number of time steps used, ε and δ correspond to equation (41). We note that δ is smaller than ε by at least an order of magnitude.

The stability domain defined in section 2.1 contains the region in $\mathbb{C}$ in which $|\mathrm{Am}(\lambda)| \leq 1$. We also denote the limit of the amplification factor as $|\lambda| \to \infty$ for $PIES_m^J$ by

$$\mu(m,J) = \lim_{|\lambda|\to\infty} \mathrm{Am}(\lambda). \tag{45}$$

The accuracy domain contains the region in which

$$|\varphi(b) - \tilde{\varphi}(b)| < \varepsilon, \tag{46}$$

where $\tilde{\varphi}$ is a numerical approximation of the solution. Since both the analytical and numerical solutions are analytical functions of λ the boundaries of the stability and accuracy domains are well defined.

The stability domain boundaries are numerically constructed via the following procedure. We start by taking a step from the origin $p_0 = (0,0)$ along the positive imaginary axis to the point p_1' and search in a small interval orthogonal to $(0,p_1')$ around p_1' for the boundary point p_1. The search is done by using the bisection method (see section 2.5). Then another step is taken to the point p_2' along the line that passes through p_1 and p_0. The procedure repeats until it intersects the real axis.

The accuracy domain boundaries are constructed as follows. The first step is taken along the positive real axis to a point p_1', and a boundary point is searched on the interval $[p_0, p_1']$. The next step is taken in the orthogonal direction to the real axis, and the subsequent steps follow exactly the same procedure as described above for the stability domain. For accuracy higher than 10^{-12}, extended precision is used. All other computations are done with double precision.

The stability and accuracy domains of the schemes is Table 1 are depicted in Figs. 1 to 6. The behavior of $PIES$ at large $|\lambda|$ is demonstrated in Fig. 7 for left and right quadrature rules. We specify the values of α and μ in the legend of Figs. 1-6.

Observations: our experiments motivate the following observations:

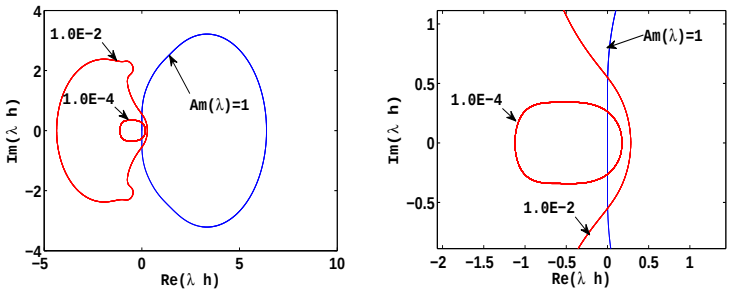

Figure 1: Left: Stability and accuracy regions for DCL1. Right: Detailed stability and accuracy regions for DCL1. $\alpha = 90°$. $\mu = 0.22$

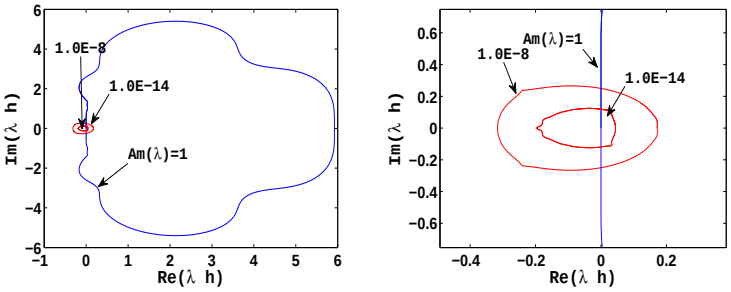

Figure 2: Left: Stability and accuracy regions for DCL2. Right: Detailed stability and accuracy regions for DCL2. $\alpha = 85°.4'$. $\mu = 0.17$

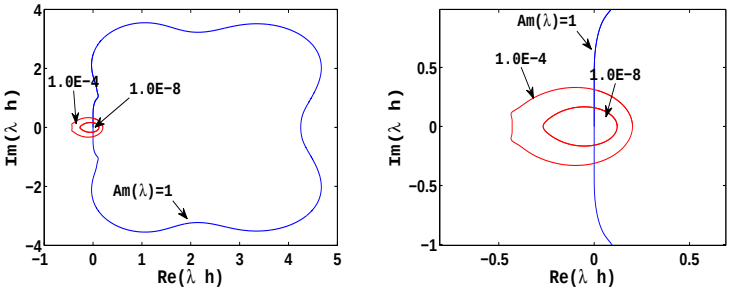

Figure 3: Left: Stability and accuracy regions for DCL2B. Right: Detailed stability and accuracy regions for DCL2B. $\alpha = 89°.1'$. $\mu = 0.16$

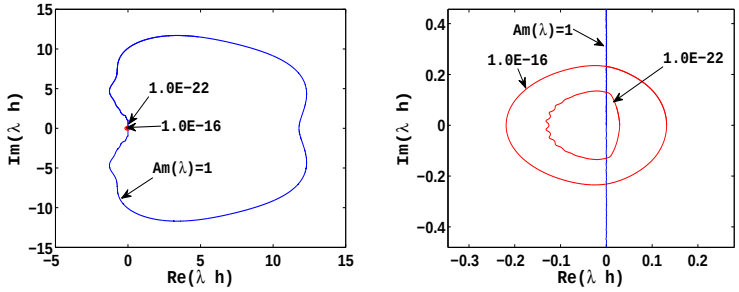

Figure 4: Left: Stability and accuracy regions for DCL3. Right: Detailed stability and accuracy regions for DCL3. $\alpha = 74°.8'$. $\mu = 0.16$

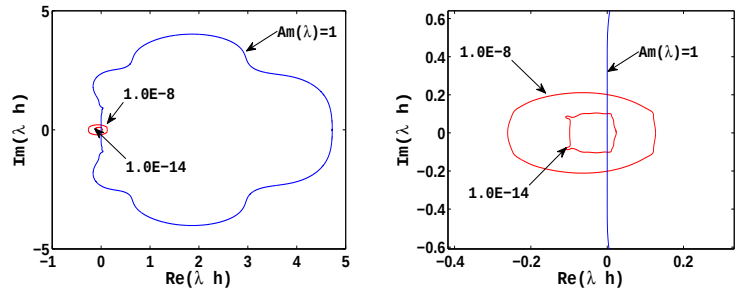

Figure 5: Left: Stability and accuracy regions for DCL4. Right: Detailed stability and accuracy regions for DCL4. $\alpha = 84°.8'$. $\mu = 0.77$

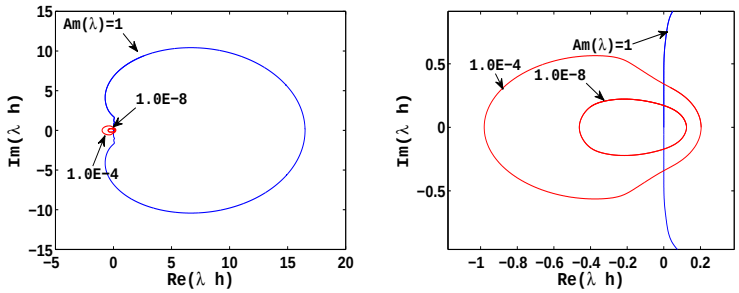

Figure 6: Left: Stability and accuracy regions for DCR2B. Right: Detailed stability and accuracy regions for DCR2B. $\alpha = 80°.18'$. $\mu = 10^{-15}$

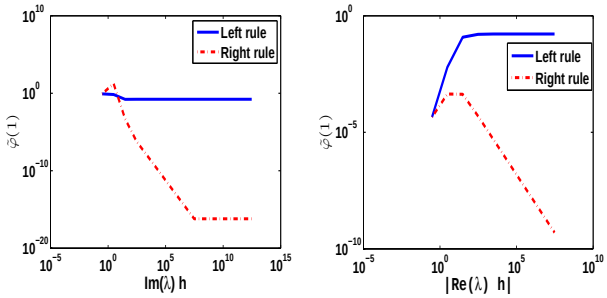

Figure 7: L-stability analysis for left and right quadrature rules for DCL2B.

Table 2: Number of Steps per Wavelength

Scheme/accuracy	2 digits	4 digits	8 digits	14 digits	16 digits	22 digits
DCL1	11	24	-	-	-	-
DCL2B	-	20	39	-	-	-
DCL2	-	-	25	52	-	-
DCL3	-	-	-	-	27	48
DCL4	-	-	31	62	-	-

1. The region of instability is within the boundaries marked by $\mathrm{Am}(\lambda) = 1$, and the domain of stability extends to infinity. The accuracy domain is within the closed accuracy curves.
2. Increasing the accuracy of the scheme (by increasing the number of iterations along with increasing the accuracy of the quadrature weights) increases the instability region.
3. All methods are A-stable with values for α that are specified in each legend. Typically, A-stability with $\alpha = 90^\circ$ prevails for up to 10th order for the DCL schemes tested.
4. The accuracy region of the method is close to the accuracy of the exponential fit: for example, DCL3 is designed to obtain accuracy of 10^{-24} within the radius $\rho = 3.15$ with $k = 52$. Fig. 4-right and Table 2 demonstrates an accuracy of 10^{-22} obtained within a radius of 0.13 (in the λh plane) which corresponds to 48 steps per wavelength. We elaborate the computed number of steps per wavelength in Table 2.
5. A comparison between DCL2 and DCL4 suggests that increasing the number of time steps while preserving the same accuracy of the exponential fit increases the stability region.

6. As seen in Fig. 7, right hand quadrature rules (rhr) are L-stable. However, for rhr-based scheme α is limited: for example DCL2B with $J = 7$ is not A-stable, and further experiments with $J > 7$ show that α grows relatively fast. Lhr-based methods are A-stable (with $\alpha = 90^\circ$) up to $J = 9$. Most important is that for all lhr-based methods $\mu(m, J) < 1$, which makes them practical for stiff problems.

6 Numerical Results

In this section the performance of $PIES$ is examined for problems of different stiffness levels and for different parameter choice. The experiments are chosen to give an informative demonstration of the solver performance and to allow a comparison with other implicit solvers (such as [5], [6], and [7]). We also present results for the adaptive implementation (section 4.2) in section 6.3. Throughout this section, our experiments employ the following three stiff problems:

1. The cosine problem

$$\varphi' = F(\varphi, t) = -2\pi sin(2\pi t) - \frac{1}{\epsilon}(\varphi - cos(2\pi t)), \quad \varphi(0) = 1. \tag{47}$$

The exact solution of (47) is

$$\varphi = cos(2\pi t)) \tag{48}$$

The cosine problem becomes increasingly stiff as $\epsilon \to 0$

2. The non-linear problem of Van der Pol (VdP) equation

$$\varphi''(t) + \mu(1 + \varphi(t)^2)\varphi'(t) + \varphi(t) = 0. \tag{49}$$

Equation (49) can be transformed to the 2×2 system

$$\begin{aligned} \varphi_1' &= \varphi_2 \\ \varphi_2' &= (-\varphi_1 + (1 - \varphi_1^2)\varphi_2)/\epsilon \end{aligned} \tag{50}$$

via the substitution $\varphi_1 = x(t)$, $\varphi_2 = \mu x'(t)$ and $t = t/\mu$, where $\epsilon = 1/\mu^2$ is a parameter controlling the stiffness. We examine the performance of $PIES$ on the interval $[0, 0.5]$ to demonstrate its efficiency for different accuracy orders and stiffness levels. We also perform experiments on the interval $[0, 2]$, in which the VdP solution has a challenging layer for small ϵ. Initial conditions for VdP equation depend on ϵ, and are specified in table 3.

Table 3: Initial conditions for VdP system

Stiffness parameter	$y_1(0)$	$y_2(0)$
$\epsilon = 10^{-1}$	2	-0.65
$\epsilon = 10^{-2}$	2	-0.6654321
$\epsilon = 10^{-3}$	2	-0.66654321
$\epsilon = 10^{-4}$	2	-0.666654321
$\epsilon = 10^{-5}$	2	-0.6666654321
$\epsilon = 10^{-6}$	2	-0.66666654321

3. The nonlinear oscillating circle problem

$$\begin{aligned} \varphi_1' &= -\varphi_2' - \epsilon\varphi_1(1 - \varphi_1^2 - \varphi_2^2) \\ \varphi_2' &= \varphi_1' - 3\epsilon\varphi_2(1 - \varphi_1^2 - \varphi_2^2), \end{aligned} \tag{51}$$

where ϵ is the stiffness parameter. This is a nonlinear oscillator along the unit circle, with other solutions rapidly spiraling towards its smooth manifold from the inner and the outer domain. For stronger stiffness, the spectrum of the Jacobian does not behave like a typical stiff spectrum in some vicinity of the manifold (because $(0, 0)$ is a strongly repelling fixed point). The exact solution of (51) $\varphi(t) = (cos(t); sin(t))$, and we solve (51) on the interval $[0, 3]$.

6.1 Non-Stiff Examples

To demonstrate the classical convergence order of $PIES$ for non-stiff problems, we numerically solve the cosine problem (47) and the VdP problem (50) with $\epsilon = 0.5$ and $\epsilon = 1$, respectively.

The cosine problem is solved on $[0, 10]$ and the error is measured by the L_2-norm of the error in time of $PIES^J_{34}(F, 0)$ from the exact solution (48). The error is plotted as a function of the number of function calls in Fig. 8-left. A similar test is performed with the VdP equations on the interval $t \in [0, 4]$. The numerical solution is computed with $PIES^J_{34}(F, 0)$ with $J = 2, 4, 6$, and a reference solution is computed by $PIES^{18}_{52}$. The absolute difference between the two solutions at $t = 4$ is plotted in Fig. 8-right for φ_1 as a function of the number of function calls. The results for y_2 are similar.

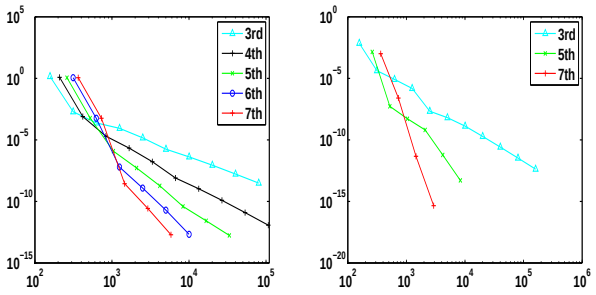

Figure 8: Error vs. number of function calls for non-stiff problems. **Left**: Cosine problem solution with $PIES^J_{34}(F, 0)$ with $J = 2, ..., 6$. **Right**: VdP solution with $PIES^J_{34}(F, 0)$ with $J = 2, 4, 6$.

6.2 Stiff Examples

The performance of the non-adaptive $PIES$ algorithm on stiff problems is illustrated in Figs. 9-12. First, we demonstrate in Fig. 9 the efficiency of $PIES$ for solving the cosine problem (47) where the error is measured by the absolute difference at $t = 10$, for $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$. In Fig. 10-left the efficiency of methods of orders 3,4,5,6, for the VdP system (50) with stiffness parameter $\epsilon = 10^{-3}$ on the interval $[0, 0.5]$, is demonstrated. In Fig. 10-right convergence results for the stiffer case - $\epsilon = 10^{-6}$ with methods of orders 5,9,13 are shown too. The errors in Fig. 10 are computed for the stiff component y_2 at $t = 0.5$ using a reference solution $PIES^{16}_{52}(F, 0)$.

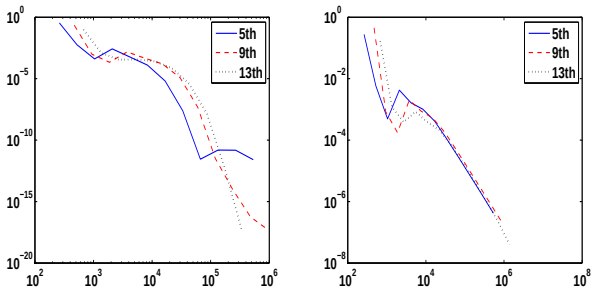

Figure 9: Error vs. number of function calls for stiff cosine problems. **Left**: $PIES^J_{52}(F, 0)$ with $J = 4, 8, 12$ for the cosine problem with $\epsilon = 10^{-3}$. **Right**: $PIES^J_{52}(F, 0)$ with $J = 4, 8, 12$ for the cosine problem with $\epsilon = 10^{-6}$.

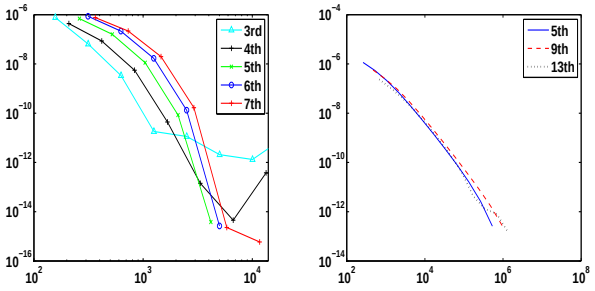

Figure 10: Error vs. number of function calls for stiff VdP problems. **Left**: $PIES^J_{52}(F,0)$ of orders $J = 3,4,5,6,7$ for the VdP problem with $\epsilon = 10^{-3}$. **Right**: $PIES^J_{52}(F,0)$ with $J = 5,9,13$ for the VdP problem with $\epsilon = 10^{-6}$.

Fig. 11 presents convergence results for varying the stiffness parameter ϵ with the 7th-order method $PIES^6_{52}$. For this experiment the error is computed for $y_2(0.5)$ using the reference solution $PIES^{16}_{52}(F,0)$.

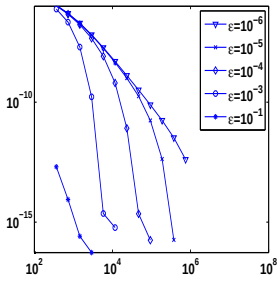

Figure 11: Error vs. number of function calls for the VdP problem with varying stiffness, for the solution $PIES^6_{52}(F,0)$.

Fig. 12 presents the global error for the solutions of the oscillating circle problem (51) on $[0,3]$ with $\epsilon = -10^3$ and with $\epsilon = -10^5$, as reported, for example, in [20] and [21].

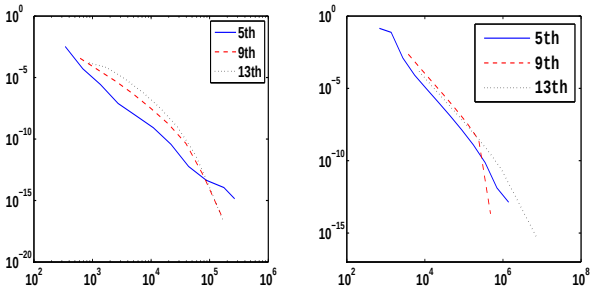

Figure 12: Error vs. number of function calls for the oscillating circle problem. **Left**: $PIES^J_{52}(F,0)$ with $J = 4,8,12$ for a problem with $\epsilon = 10^{-3}$. **Right**: $PIES^J_{52}(F,0)$ with $J = 4,8,12$ for a problem with $\epsilon = 10^{-5}$.

Table 4: Number of function calls for adaptive and non-adaptive codes for $PIES_{34}^{10}$, with accuracy $\varepsilon = 10^{-9}$

ϵ	Adaptive	Non-adaptive
$\epsilon = 10^{-1}$	7582	6000
$\epsilon = 10^{-2}$	24K	51K
$\epsilon = 10^{-3}$	52K	450K
$\epsilon = 10^{-4}$	112K	5625K
$\epsilon = 10^{-5}$	176K	52M

6.3 Adaptive Implementation

We demonstrate the use of the adaptive scheme described in section 4. For this demonstration the VdP system (50) is solved on the interval $[0, 2]$ in which the solution contains a challenging layer. Different stiffness parameter (ϵ) values are examined, so that as $\epsilon \to 0$ the problem is stiffer and the layer is steeper. The efficiency of $PIES_{34}^{10}$ with uniform step size is compared with the adaptive code for accuracy $\varepsilon = 10^{-9}$. The error is measured at $t = 2$ using a reference solution computed with the adaptive code and $PIES_{34}^{12}$ to accuracy $\varepsilon = 10^{-12}$. The comparison is given in Table 4.

6.4 Observations

1. Classical convergence is demonstrated for all methods for the non-stiff case. The performance demonstrated in Fig. 8 is similar to the performance of the solver in [7] on the cosine problem (47), and to the performance of the solver in [6] for the VdP problem (50).

2. For the moderately stiff ($\epsilon = 10^{-3}$) cosine problem, high convergence is observed only for sufficiently small time steps, the same observation holds for the VdP problem. In particular, the method of order 13 converges with 9th-order (that is by all means sufficient for our needs). For very stiff problems ($\epsilon = 10^{-6}$) only second order convergence is observed for methods of all orders tested. Our observations are reinforced again by the results demonstrated in Fig. 11 for varying the stiffness parameter ϵ. This phenomenon, regarded as *order reduction* for stiff problems, is evident and discussed in detail in the literature (see for example [5], [12]). The oscillating circle example manifests order reduction on both moderate and high stiffness. In particulary we observe reduction to fifth and fourth order, respectively.

3. Our results compare favorably with the results of deferred correction methods suggested in [6] (see Figs. 5.4 and 5.5 therein). For example, for the VdP problem with moderate stiffness, a 7th order method obtains accuracy of 10^{-12} with 10^4 function calls, whereas the same accuracy is obtained by $PIES_{52}^{6}$ with $4 \cdot 10^{-3}$ function calls.

4. Our results also compare favorably with the results reported in [7] (see Fig. 5.6 therein) where for the VdP problem with $\epsilon = 10^{-3}$ an accuracy of 10^{-12} is obtained by $2 \cdot 10^4$, by using either Gauss-Legendre, Gauss-Radau, Gauss-Lobatto or uniform quadrature nodes. For $\epsilon = 10^{-6}$ $PIES$ efficiency is inferior when compared with the deferred correction

methods using Gauss-Legendre, Gauss-Radau, or Gauss-Lobatto [7]: we obtain accuracy of 10^{-12} using $5 \cdot 10^5$ function calls, while in [7] for the above discretizations this accuracy is obtained in the range of $10^4 - 10^5$ function calls. However, it can be seen that in this regime these methods demonstrate reduction to first order convergence while $PIES$ reduces to second order. Most noticeable, is that the convergence reported in [7] for a uniform discretization stalls for $\epsilon = 10^{-5}$ and for $\epsilon = 10^{-6}$ even when step-size is decreased. $PIES$ keeps converging with 2nd-order to machine precision.

We note that the experiments with deferred correction presented in [6] and [7] use the ladder method which suggests adaptively increasing the number of quadrature nodes (and their overall accuracy) as the iterations continue. Such a scheme is not implemented in our method and will further improve the efficiency of $PIES$.

5. Our results also compare favorably with the additive Runge-Kutta (ARK) schemes described in [5]. For example, in the moderately stiff case a 4th order ARK method achieves accuracy of 10^{-7} with 10^3 function calls (see Figs. 8 and 9 therein), whereas our method achieves accuracy 10^{-9} for the same number of function calls. For the stiffer case the efficiency of both $PIES$ and ARK is approximately the same for the range of function evaluations reported (see [7], Fig. 5.5 therein). However, the reduction of order of $PIES$ is to second order, while the reduction of order for ARK is to a first order. Thus, for smaller step sizes we conclude that $PIES$ can obtain higher accuracy than ARK.

6. The adaptive implementation is crucial for solving stiff equations in which the solution is geometrically challenging to approximate (e.g. layers), and stiffness is high. The efficiency is by orders of magnitude better than in the uniform step-size solver, which in highly stiff cases is inefficient.

7 Conclusion and Future Work

We have described a Picard integral equation solver for stiff ordinary differential equations. Our solver is based on an approximation of the solution as a linear combination of exponentials on a uniformly discretized grid. The stability and accuracy properties of the solver, as well as numerical experience suggest that $PIES$ is a competitive alternative to other deferred correction methods as well as for Runge-Kutta solvers. Future work entails the integration of high order schemes such as Runge-Kutta, as well as predictor corrector schemes for stiff problems. Applications to partial differential equations will be considered as well.

8 Acknowledgment

We thank the associate editor and the anonymous reviewers for their insightful comments which improved our manuscript.

References

[1] D A. Dutt, L. Greengard, and V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT, 40(2):241266, 2000.12

[2] R. Frank and C. Ueberhuber, Iterated deferrred correction for the efficient solution of stiff systems of ordinary differential equations, BIT 17 (1977), 146-159.

[3] R. Frank and C. Ueberhuber, Iterated deferred correction for differential equations, part 1, Computing, 20 (1978), 207-228.

[4] K. Böhmer and H. J. Stetter, eds., Defect correction methods, theory and applications, Springer-Verlag, New York, 1984.

[5] C. A. Kennedy and M. H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Appl. Numer. Math., 44:139181, 2003.

[6] M. L. Minion. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. Appl. Numer. Math., 48(3-4):369 387, 2004.

[7] A. T. Layton and M. L. Minion, Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations, BIT 45:2 (2005), 341373.

[8] A. T. Layton, M. L. Minion, Implications of the choice of predictors for semi-implicit picard integral deferred correction methods Comm. App. Math. And Comp. Sci. 2(1), (2007)

[9] J. Huang, J. Jia, M. L. Minion, Accelerating the convergence of spectral deferred correction methods, J. comput. Phys., 214(2), 633-656, (2006).

[10] A. J. Christlieb, B. Ong, J. Qiu: Integral Deferred Correction Methods Constructed with High Order Runge-Kutta Methods, to appear in Mathematics of Computation

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

[12] E. Hairer and G.Wanner, Solving ordinary differential equations II, Stiff and Differential Problems, Springer, 1993.

[13] 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. 6378.

[14] W. Gautschi, Numerical integration of ordinary differential equations based on trigonometric polynomials, Numerische Mathematik, 3 (1961), pp. 381397.

[15] M. Mäkelä, O. Nevanlinna, A.H. Sipilä, Exponentialy fitted multistep methods by generalized Hermite-Birkhoff interpolation, BIT Numer. Math. 14, 437-451 (1974).

[16] A. Glaser, V. Rokhlin, A new class of highly accurate solvers for ordinary differential equations, Journal of Scientific Computing, Volume 38:3, 368 - 399 (2009).

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

[18] H. Cheng, Z. Gimbutas, P. Martinsson, and V. Rokhlin, On the compression of low rank matrices, SIAM Journal on Scientific Computation, 26 (2005), pp. 13891404.

[19] Germund Dahlquist, Åke Björck, Numerical nethods, Courier Dover, 2003.

[20] W. Auzinger, H. Hofstätter, W.Kreuzer, E.Weinmüller: Modified defect correction algorithms for ODEs. Part I: General theory, Numer. Algorithms, vol. 36:135 2004.

[21] W. Auzinger, H. Hofstätter, W.Kreuzer, E.Weinmüller: Modified defect correction algorithms for ODEs. Part II: Stiff initial value problems, Numer. Algorithms 40, 285303, 2005.