BIT xx (19xx), xxx–xxx.

Spectral Deferred Correction Methods for Ordinary Differential Equations *

Alok Dutt† Leslie Greengard‡ Vladimir Rokhlin§

Abstract.

We introduce a new class of methods for the Cauchy problem for ordinary differential equations (ODEs). We begin by converting the original ODE into the corresponding Picard equation and apply a deferred correction procedure in the integral formulation, driven by either the explicit or the implicit Euler marching scheme. The approach results in algorithms of essentially arbitrary order accuracy for both non-stiff and stiff problems; their performance is illustrated with several numerical examples. For non-stiff problems, the stability behavior of the obtained explicit schemes is very satisfactory and algorithms with orders between 8 and 20 should be competitive with the best existing ones. In our preliminary experiments with stiff problems, a simple adaptive implementation of the method demonstrates performance comparable to that of a state-of-the-art extrapolation code (at least, at moderate to high precision).

Deferred correction methods based on the Picard equation appear to be promising candidates for further investigation.

AMS subject classification: 65F20.

Key words: Spectral methods, initial value problems, deferred correction, stiffness.

1 Introduction

The construction of efficient, stable and high order methods for solving initial value problems governed by systems of ordinary differential equations (ODEs) is, in many respects, a mature subject [1, 2, 7, 10, 11, 15]. Existing methods for such problems can be classified, coarsely speaking, into two groups. The first group consists of intrinsically high-order discretization schemes (Runge-Kutta methods, linear multistep methods, etc.) and the second group consists of methods based on accelerating the convergence of low-order schemes through the use of Richardson extrapolation or deferred correction. For non-stiff problems, there exist extremely effective discretizations of order up to twelve or so, at which point the stability constraints imposed on the schemes become too severe. From

*Received September 1998.

†† Bank of America, 1 Alie Street, London E1 8DE, England

‡Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. The work of this author was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DEFGO288ER25053.

§Departments of Mathematics and Computer Science, Yale University, New Haven, CT 06520. The work of this author was supported by DARPA/AFOSR under Contract F49620-95-C-0075 and by ONR under Grant N00014-96-1-0188.

that point on, most practitioners recommend extrapolation. For stiff problems, the situation is considerably more complicated. Implicit Runge-Kutta methods possess excellent stability properties, but are very expensive when high order accuracy is required ("singly-implicit" methods are an exception [2]). Implicit multistep algorithms can have very high order convergence, but tend to have relatively poor stability properties. Most practitioners, therefore, recommend some form of Runge-Kutta (or backward differentiation) method for orders up to five or so, and again turn to extrapolation when higher order accuracy is needed. These extrapolation methods, while effective, are still expensive, since they require computing a sequence of solutions on finer and finer grids. Although deferred correction approaches also require computing a sequence of solutions, they are more efficient in theory; the convergence rate can be made to increase more rapidly and the same underlying grid is used on each sweep. Because of various numerical instabilities, however, their use has generally been limited to the conversion of second-order accurate solutions into fourth or sixth-order accurate ones.

In this paper, we present a new version of the deferred correction approach. It is based on replacing the original ODE with the corresponding Picard integral equation and discretizing the interval on which the ODE is to be solved into a composite Gauss-Legendre grid. We then solve the integral equation approximately with either the explicit Euler method (for non-stiff problems) or the implicit Euler method (for stiff problems) and correct the solution to higher and higher order accuracy by solving a sequence of "error" equations on the same grid with the same marching scheme. Because we use spectral integration [8, 9], we refer to this class of methods as *spectral deferred correction* (SDC) methods. For non-stiff problems, the approach results in algorithms of essentially arbitrary order accuracy. Moreover, as can be seen in section 5.1 below, the stability behavior of the resulting schemes is very satisfactory. Our preliminary tests indicate that the schemes with orders between 8 and 20 are roughly competitive with the best existing ones. Our principal goal, however, is the stiff case, especially in the environment where high precision is required. Since our stiff schemes are driven by the implicit Euler method, we do have to solve systems of (generally) non-linear equations at each time step; unlike general implicit Runge-Kutta techniques, however, we do not need to solve systems of equations whose dimensionality is greater than that of the underlying ODE.

In certain respects, this paper should be viewed as an experimental one. While the convergence rates of the techniques we present are easily proven, their stability properties are established numerically. For orders up to 5, we have obtained schemes that are both L-stable and A-stable. For orders up to 30, we have obtained schemes that are L-stable and $A(\alpha)$-stable, with α extremely close to $90°$ (see section 5.2 below). We have no analytical reason to believe that A-stable schemes of order 6 or higher do not, in fact, exist.

To fix notation, we assume that the initial value problem to be solved is in the standard form

$$\varphi'(t) = F(t, \varphi(t)) \qquad t \in [a, b] \tag{1.1}$$

$$\varphi(a) = \phi_a \ , \tag{1.2}$$

where $\varphi_a, \varphi(t) \in \mathbb{C}^n$ and $F : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$. Requiring that $F \in C^1(\mathbb{R} \times \mathbb{C}^n)$ is, of course, sufficient to guarantee local existence and uniqueness of the problem (1.1), (1.2). Since we are interested in high-order methods, however, we suppose throughout that F is sufficiently smooth. Unless otherwise stated, we assume that the dimension of the system $n = 1$, since it makes much of the discussion less cumbersome.

The structure of this paper is as follows. In section 2, we introduce several analytical and numerical prerequisites, in section 3, we describe the classical deferred correction scheme and the difficulties it encounters, and in section 4, we describe the new spectral deferred correction approach. In section 5, we investigate the stability and accuracy properties of a variety of SDC schemes and in section 7, we illustrate the performance of these schemes with several examples. In section 8, we discuss possible extensions and generalizations of the approach.

2 Analytical and Numerical Preliminaries

In this section, we summarize several well-known facts from numerical analysis. First, suppose that we are solving the problem (1.1), (1.2) numerically on the interval $[a, b]$, obtaining an approximate solution $\widetilde{\varphi}(b)$. The two critical characteristics of such a scheme with which we are concerned here are its order of accuracy and its (stiff) stability. A numerical method is said to be of *order of accuracy* or *order k* if, for any sufficiently smooth F, there exists a real constant $K > 0$ such that

$$\|\widetilde{\varphi}(b) - \varphi(b)\| < K \cdot (b-a)^{k+1}. \tag{2.1}$$

The suitability of a numerical method for stiff problems is generally analyzed by applying it to the equation

$$\begin{aligned} \varphi'(t) &= \lambda \cdot \varphi(t) \qquad t \in [0,1] \\ \varphi(0) &= 1 \ , \end{aligned} \tag{2.2}$$

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

$$Am(\lambda) = \widetilde{\varphi}(1). \tag{2.3}$$

If, for a given value of λ,

$$| \ Am(\lambda) \ | \leq 1, \tag{2.4}$$

then the numerical method is said to be stable for that value of λ. If a method is stable for all λ in the left-half plane ($Re(\lambda) \leq 0$), 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

$$\lim_{Re(\lambda) \to -\infty} Am(\lambda) = 0. \tag{2.5}$$

2.1 The Picard Integral Equation

Integrating equations (1.1) and (1.2) with respect to t, we obtain the equivalent Picard equation

$$\varphi(t) = \varphi_a + \int_a^t F(\tau, \varphi(\tau))d\tau. \tag{2.6}$$

Suppose now that we have obtained an approximate solution $\varphi^0(t)$ to (2.6). A measure of the quality of the approximation is given by the residual function

$$\varepsilon(t) = \varphi_a + \int_a^t F(s, \varphi^0(s))ds - \varphi^0(t). \tag{2.7}$$

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

$$\delta(t) = \varphi(t) - \varphi^0(t). \tag{2.8}$$

Substituting (2.8) back into (2.6), we obtain

$$\varphi^0(t) + \delta(t) = \varphi_a + \int_a^t F(s, \varphi^0(s) + \delta(s))ds, \tag{2.9}$$

or, after some algebraic manipulation,

$$\delta(t) = \int_a^t \left[F(s, \varphi^0(s) + \delta(s)) - F(s, \varphi^0(s))\right] ds + \varepsilon(t). \tag{2.10}$$

Letting the function $G : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ be given by

$$G(t, \delta) = F(t, \varphi^0(t) + \delta(t)) - F(t, \varphi^0(t)), \tag{2.11}$$

we can rewrite (2.10) in the form

$$\delta(t) - \int_a^t G(s, \delta(s))ds = \varepsilon(t), \tag{2.12}$$

which is a Picard-type integral equation like (2.6).

2.2 Euler Methods for the Picard Equation

Suppose that $t_0, t_1, t_2, \cdots, t_m, t_{m+1}$ is a refinement of the interval $[a, b]$ with

$$t_0 = a, \tag{2.13}$$
$$t_{m+1} = b, \tag{2.14}$$
$$t_0 < t_1 < t_2 \quad \cdots \quad < t_m < t_{m+1}. \tag{2.15}$$

Then, the explicit Euler (or forward Euler) method for the solution of the ODE (1.1) or the integral equation (2.6) is given by the formula

$$\varphi_{i+1} = \varphi_i + h_i \cdot F(t_i, \varphi_i) \tag{2.16}$$
$$h_i = t_{i+1} - t_i, \tag{2.17}$$

for $i = 0, 1, \cdots, m$. The implicit (or backward) Euler scheme for the solution of (1.1) is given by the formula

$$\varphi_{i+1} = \varphi_i + h_i \cdot F(t_{i+1}, \varphi_{i+1}). \tag{2.18}$$

Similarly, the explicit Euler method for the solution of (2.12) is given by

$$\delta_{i+1} = \delta_i + h_i \cdot G(t_i, \delta_i) + (\varepsilon(t_{i+1}) - \varepsilon(t_i)), \tag{2.19}$$

and the implicit Euler scheme for the solution of (2.12) by

$$\delta_{i+1} = \delta_i + h_i \cdot G(t_{i+1}, \delta_{i+1}) + (\varepsilon(t_{i+1}) - \varepsilon(t_i)). \tag{2.20}$$

Definition 2.1. *Given the function $G : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ defined in (2.11) and the vector of data $\varepsilon = \{\varepsilon(t_1), \varepsilon(t_2), \cdots \varepsilon(t_m)\} \in \mathbb{C}^m$, we define the map $C_{exp} : C^1(\mathbb{R} \times \mathbb{C}) \times \mathbb{C}^m \to \mathbb{C}^m$ by*

$$C_{exp}(G, \varepsilon) = \delta,$$

where $\delta = (\delta_1, \delta_2, \cdots, \delta_m)$ is the vector of corrections produced by the scheme (2.19). Similarly, we define the map $C_{imp} : C^1(\mathbb{R} \times \mathbb{C}) \times \mathbb{C}^m \to \mathbb{C}^m$ by

$$C_{imp}(G, \varepsilon) = \delta,$$

where $\delta = (\delta_1, \delta_2, \cdots, \delta_m)$ is the vector of corrections produced by the scheme (2.20).

A full description of the marching schemes corresponding to C_{exp} and C_{imp} requires that we specify how the values $\varepsilon(t_i)$ are actually computed. For this, we will require stable and high-order accurate methods for interpolation and integration.

2.3 Spectral Integration, Differentiation, and Interpolation

Given a natural number m, we will denote by $r_1, r_2, \cdots, r_m$ the m Gauss-Legendre nodes on the interval $[-1, 1]$ (see, for example, [19]). For an interval $[a, b] \subset \mathbb{R}$, we will denote by $s_1, s_2, \cdots, s_m$ the m Gaussian nodes on the interval $[a, b]$, given by the formula

$$s_i = \frac{b-a}{2} \cdot r_i + \frac{b+a}{2}. \tag{2.21}$$

Suppose now that $t_1, t_2, \cdots t_m$ is a strictly increasing sequence of points in $\mathbb{R}$, and that with each point t_i is associated a function value φ_i. Let $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_m)$. Then, for any point $t \in \mathbb{R}$, we will denote by $L^m : \mathbb{C}^n \times \mathbb{R} \to \mathbb{C}$ the usual Lagrange interpolant defined by the formula

$$L^m(\varphi, t) = \sum_{i=1}^{m} c_i(t) \cdot \varphi_i, \tag{2.22}$$

where the functions $c_i(t)$ are given by

$$c_i(t) = \prod_{j \neq i} \frac{t - t_j}{t_i - t_j}. \tag{2.23}$$

DEFINITION 2.2.
Let $F : \mathbb{R} \to \mathbb{C}$ and let the vector $f = \{f_1, f_2, \cdots f_m\}$ be defined by the formula

$$f_i = F(t_i). \tag{2.24}$$

If $e = \{e_1, e_2, \cdots, e_m\}$ is defined by

$$e_i = \frac{d}{dt} L^m(f, t_i)\,, \tag{2.25}$$

then the linear mapping $D^m : \mathbb{C}^m \to \mathbb{C}^m$ for which

$$e = D^m(f) \tag{2.26}$$

will be referred to as the differentiation *matrix. If $g = \{g_0, g_1, \cdots, g_m\}$ is defined by*

$$g_i = \int_{-1}^{t_i} L^m(f, t)\ dt, \tag{2.27}$$

then the linear mapping $S^m : \mathbb{C}^m \to \mathbb{C}^m$ for which

$$g = S^m(f) \tag{2.28}$$

will be referred to as the integration *matrix (see also [10], p. 212).*

If the function F is a polynomial of degree $m - 1$, and the vector f is defined as in (2.24), then

$$F(t) = L^m(f, t), \tag{2.29}$$

and the operators D^m and S^m are exact.

REMARK 2.1. The formulae (2.26) and (2.28) are not numerically stable, unless the points $t_1, t_2, \cdots t_n$ are chosen with some care; using equispaced nodes, for example, leads to the well-known Runge phenomenon. On the other hand, with a suitable choice of nodes, the operators defined in (2.26) and (2.28) become extremely effective numerical tools. The most popular choices are the Chebychev and Gauss-Legendre nodes, for which the matrices D^m, S^m are usually referred to as *spectral differentiation* and *spectral integration* matrices, respectively. We refer the reader to [8, 9, 19] for a detailed discussion of their numerical properties. Here, we simply observe that spectral integration is an extremely useful tool; the maximum eigenvalue of S^m is bounded, and its minimum eigenvalue is of the order $O(1/m^2)$. Spectral differentiation, while widely used, is somewhat limited by the fact that the maximum eigenvalue of D^m is of the order $O(m^2)$, which renders the operator ill-conditioned.

REMARK 2.2. It is easy to see that the matrices S^m, D^m are dense for any distribution of points $t_1, t_2, \cdots t_m$. Since applying a dense $m \times m$ matrix to a vector requires m^2 operations, the procedure can become quite expensive. When $t_1, t_2, \cdots t_m$ are Chebychev nodes on the interval $[a, b]$, however, the Fast Fourier Transform (FFT) can be used to apply the matrices S^m, D^m to arbitrary vectors using $O(m \log m)$ operations. For more general point distributions, a somewhat less efficient $O(m \log m)$ scheme for the application of the matrices S^m and D^m

to arbitrary vectors can be found in [5]. In the present paper, we will be using relatively short sequences ($m \approx 16$), so that the cost of applying S^m and D^m to arbitrary vectors will not be a major issue.

In section 6, we will need one more numerical tool. Let $r_1, r_2, \cdots, r_m$ denote the Gauss-Legendre nodes on the interval $[-1, 1]$. We then define the $m \times m$ matrix V^m by the formula

$$V^m_{i,j} = P_{j-1}(r_i), \tag{2.30}$$

where P_j denotes the Legendre polynomial of order j. Note that V^m maps the vector $(\alpha_1, \ldots, \alpha_m)$ into the vector $(f_1, f_2, \ldots, f_m)$, where

$$f_i = \sum_{j=1}^{m} \alpha_j P_{j-1}(r_i).$$

The matrix V^m is non-singular [8] and we will denote its inverse by

$$W^m = (V^m)^{-1}. \tag{2.31}$$

Given a polynomial Q of degree $m-1$, it is clear that the matrix W^m maps the values of Q at the nodes $r_1, r_2, \cdots, r_m$ into the coefficients of its Legendre expansion.

3 Classical Deferred Correction

Suppose now that we define a grid on the interval $[a, b]$ with $(m+1)$ equispaced nodes t_i given by

$$t_i = a + i \cdot h \qquad\qquad i = 0, ..., m \ , \tag{3.1}$$

where $h = (b-a)/m$ is the step size and that we wish to solve the ordinary differential equation (1.1), (1.2) on this grid. A kth order accurate method will yield an approximate solution $\eta = (\eta_1, \ldots, \eta_m)$ with

$$\eta_i = \varphi(t_i) + O(h^k) \ . \tag{3.2}$$

This defines the unique mth order polynomial $L^m(\eta, t)$ which interpolates the discrete approximate solution values η_i at the designated grid points t_i. We can then define an error function

$$\delta(t) = \varphi(t) - L^m(\eta, t) \tag{3.3}$$

which clearly satisfies the differential equation

$$\begin{aligned} \delta'(t) &= \varphi'(t) - \frac{d}{dt} L^m(\eta, t) \\ &= f(t, \delta(t) + L^m(\eta, t)) - \frac{d}{dt} L^m(\eta, t) && (3.4) \\ \delta(0) &= 0 \ . && (3.5) \end{aligned}$$

We can now solve this equation for the error function by the same kth order method as used for the original problem. In other words, we generate a sequence of values

$$\pi_i \approx \delta(t_i) \qquad i = 1, ..., m \ , \tag{3.6}$$

on the same grid as used previously. It is well-known that the "corrected" approximation

$$\eta_i + \pi_i \approx y(t_i) \quad i = 1, \cdots, m \tag{3.7}$$

is of $(2k)^{th}$ order accuracy [3, 6, 16, 17, 18, 21].

Iterated deferred correction proceeds by computing a new polynomial interpolant to the updated approximate grid values $(t_i, \eta_i + \pi_i)$, defining a new error function, and solving a new correction equation of the same form as (3.4) above.

Algorithm: Deferred Correction

Comment [Compute initial approximation]

Using a k^{th} order method, compute an approximate solution $\varphi_i^{[0]} \approx \varphi(t_i)$ at the grid points $t_i, \ i = 1, ..., m$ on the interval $[0, T]$.

Comment [Compute successive corrections.]

do j = 1,. . . ,J

1) Compute the interpolating polynomial $L^m(\varphi^{[j-1]}, t)$.

2) Define the error function $\delta(t) = \varphi(t) - L^m(\varphi^{[j-1]}, t)$.

3) Form the error equation $\delta'(t) = f(t, \delta(t) + L^m(\varphi^{[j-1]}, t)) - \frac{d}{dt} L^m(\varphi^{[j-1]}, t)$ with $\delta(0) = 0$.

Comment [Note that the values of the derivative $\frac{d}{dt} L^m(\varphi^{[j-1]}, t)$ at the grid points are contained in the vector $D^m \varphi^{[j-1]}$, where D^m is the differentiation matrix.]

4) Using a k^{th} order method, compute an approximate solution $\pi_i \approx \delta(t_i)$ at the grid points t_i on the interval $[0, T]$.

5) Define a new approximate solution $\varphi_i^{[j]} = \varphi_i^{[j-1]} + \pi_i$.

enddo

At the end of this procedure, the error is of the order

$$O(h^{(J+1) \cdot k}) \ . \tag{3.8}$$

Of course, this process can only be repeated so long as $L^m(\varphi^{[j]}, t)$ and $\frac{d}{dt} L^m(\varphi^{[j]}, t)$ are sufficiently accurate. The usual estimate of the order of accuracy obtained with iterated deferred correction is [3]

$$O(h^{\min[(J+1) \cdot k, m]}) \ . \tag{3.9}$$

There are two independent factors which prevent the use of large m, and which have prevented large numbers of iterations from being used in practice. The first problem relates to the instability of approximation at equispaced nodes; as mentioned earlier, the procedure is numerically ill-conditioned (the Runge phenomenon). The second problem is that the procedure involves numerical differentiation in the construction of the new right-hand side for each error equation. Differentiation introduces subtle instabilities which prevent the effective use of large m (for related phenomena, see [12, 20]). The difficulty of interpolation is easily eliminated through the use of Legendre polynomials. The need for numerical differentiation is eliminated by using the Picard equation as our starting point.

4 Spectral Deferred Correction schemes

Suppose now that we are given an approximate solution $\varphi^{[0]}$ on the interval $[a, b]$. We have already described the error equation (2.12) which arises from the Picard formulation of the original ordinary differential equation, and it remains only to complete our description of the discretization process.

In the remainder of this paper, we will use the grid $s_1, \ldots, s_m$, corresponding to the standard Gauss-Legendre nodes on $[a, b]$. $\varphi^{[j]}$ will be used to denote the jth approximate solution

$$\varphi^{[j]} = (\varphi_1^{[j]}, \varphi_2^{[j]}, \ldots, \varphi_m^{[j]}) \approx (\varphi(s_1), \varphi(s_2), \ldots, \varphi(s_m)),$$

$\overline{\varphi}$ will be used to denote the m-vector $(\varphi_a, \varphi_a, \ldots, \varphi_a)$, and $\overline{F}(\varphi^{[j]})$ will be used to denote the vector

$$(F(s_1, \varphi^{[j]}(s_1)), F(s_2, \varphi^{[j]}(s_2)), \ldots F(s_m, \varphi^{[j]}(s_m))).$$

The residual function $\varepsilon(t)$ defined in (2.7) will be approximated by the vector $\sigma(\varphi^{[j]})$ defined by

$$\sigma(\varphi^{[j]}) = S^m \, \overline{F}(\varphi^{[j]}) - \varphi^{[j]} + \overline{\varphi_a}. \tag{4.1}$$

Observe that (4.1) is obtained from (2.7) by replacing exact integration with spectral integration. We may now proceed with the construct of high order schemes for both stiff and nonstiff ODEs.

Algorithm: Spectral Deferred Correction

Comment [Compute initial approximation]

For nonstiff/stiff problems, use the forward/backward Euler method to compute an approximate solution $\varphi_i^{[0]} \approx \varphi(s_i)$ at the nodes $s_1, \ldots, s_m$ on the interval $[a, b]$.

Comment [Compute successive corrections.]

do j = 1,. . . ,J

1) Compute the approximate residual function $\sigma(\varphi^{[j-1]})$.

2a) For nonstiff problems, compute $\delta^{[j]} = C_{exp}(G, \sigma(\varphi^{[j-1]}))$ as in section 2.2.

2b) For stiff problems, compute $\delta^{[j]} = C_{imp}(G, \sigma(\varphi^{[j-1]}))$ as in section 2.2.

3) Update the approximate solution $\varphi^{[j]} = \varphi^{[j-1]} + \delta^{[j]}$.

enddo

DEFINITION 4.1. *For nonstiff problems, the numerical method outlined in the preceding algorithm using m nodes and J correction steps will be denoted by $EuExp_m^J$. The approximate solution $\varphi^{[J]}$ generated by the scheme will be denoted by $EuExp_m^J(F, \varphi_a)$. For stiff problems, the numerical method outlined in the preceding algorithm using m nodes and J correction steps will be denoted by $EuImp_m^J$. The approximate solution $\varphi^{[J]}$ generated by the scheme will be denoted by $EuImp_m^J(F, \varphi_a)$.*

As in the classical deferred correction case, it is straightforward to obtain the following result [3].

THEOREM 4.1.
For any sufficiently smooth function $F : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ and any natural numbers m, k, each of the approximations $EuExp_m^J(F, \varphi_a)$ and $EuImp_m^J(F, \varphi_a)$ converge to the exact solution $(\varphi_(s_1), \ldots, \varphi(s_m))$ with order of accuracy $\min(m, J+1)$.

REMARK 4.1. An apparent drawback of the schemes $EuExp_m^J$ and $EuImp_m^J$ as tools for the solution of the initial value problem (1.1), (1.2) on the interval $[a, b]$ is the fact that the nodes $s_1, \ldots, s_m$ lie inside the interval, so that no solution is generated at the endpoint b. This problem is easily remedied by using the interpolating polynomial to obtain

$$\varphi_b = L^m(EuExp_m^J(F, \varphi_a), b). \tag{4.2}$$

If the solution is desired at an arbitrary point t in the interval $[a, b]$ we again use the Lagrange interpolant $L^m(EuExp_m^J(F, \varphi_a), t)$.

REMARK 4.2. **(Systems of ODEs)** When considering systems of ODEs, one simply performs the interpolation and integration operations componentwise. The function evaluation and/or inversion which is required at each step of (2.19) or (2.20) is obviously more complicated, but has no effect on the overall structure of the schemes $EuExp_m^J$ and $EuImp_m^J$.

REMARK 4.3. In the stiff case, a general implementation of (2.20) will involve the solution of a nonlinear equation (or, more generally, a system of nonlinear equations). Normally, this is done using some form of Newton's method (see, for example, [13]). One is then confronted with numerous issues such as the choice of the initial approximation, error control, iteration count, and the frequency with which the Jacobian of G is recomputed and inverted. In the context of (2.20), most of these issues are simple (except when evaluating the initial approximation $\varphi^{[0]}$). Indeed, since the correction δ is expected to be small, 0 can be chosen as the initial approximation, only one iteration of the Newton procedure is required at each step, and the recomputation and inversion of the Jacobian can be bypassed once the accuracy of the approximation is of the order $\sqrt{\varepsilon}$, where ε is the desired accuracy of the computation.

4.1 Behavior at Large $|\lambda|$ and Extrapolation

While the methods $EuExp_m^J$ and $EuImp_m^J$ tend to be quite satisfactory for non-stiff and mildly stiff problems, for strongly stiff problems we will need an additional modification. We start with the following obvious theorem.

THEOREM 4.2.

For any pair of natural numbers m, J, the amplification factor $Am(\lambda)$ associated with the scheme $EuImp_m^J$ is a rational function of λ. Furthermore, there exists a real number $\mu(m, J)$ such that

$$\lim_{|\lambda|\to\infty} Am(\lambda) = \mu(m, J). \tag{4.3}$$

For all combinations m, J we have tested, $\mu(m, J) < 1$, making them acceptable for stiff problems. We have not encountered any combinations m, J for which the scheme $EuImp_m^J$ is L-stable, though some are $A(\alpha)$-stable with fairly large α. Fortunately, the above theorem provides a mechanism for combining two different schemes with different m, J to obtain L-stable schemes; for reasons the authors do not completely understand, the resulting schemes also tend to have much improved $A(\alpha)$ - stability.

COROLLARY 4.3.

Suppose that m_1, j_1, m_2, j_2 are four positive integers such that $\mu(m_1, j_1) \neq \mu(m_2, j_2)$, and $EuComb_{m_1,m_2}^{j_1,j_2}$ is the scheme for the solution of the problem (1.1), (1.2) defined by the formula

$$EuComb_{m_1,m_2}^{j_1,j_2} = \frac{\mu(m_1, j_1) \cdot EuImp_{m_2}^{j_2} - \mu(m_2, j_2) \cdot EuImp_{m_1}^{j_1}}{\mu(m_1, j_1) - \mu(m_2, j_2)}. \tag{4.4}$$

Then $EuComb_{m_1,m_2}^{j_1,j_2}$ is L-stable.

REMARK 4.4. We have not carried out a systematic investigation of the properties of spectral deferred correction schemes based on Gauss-Radau or Gauss-Lobatto discretization, which includes one or both endpoints. We have, however, experimented with Chebychev discretization in this context, and obtained results very similar to those reported in this paper. The highest order $EuComb$ scheme which we found to be A-stable scheme appears to be 3, whereas with Gaussian nodes, $EuComb_{6,5}^{5,5}$ is A-stable and has order 5. We have no analytical results explaining this difference, and for most practical purposes, Gaussian-based and Chebychev-based schemes are very similar. Because of this difference, we conjecture that there may exist nodes which lead to A-stable schemes of order higher than 5.

4.2 Composite Schemes and Stability issues

When considering a general purpose solver for the initial value problem (1.1), (1.2) on the interval $[a, b]$, it is rarely reasonable to use a single global mesh. Thus, we assume that the interval $[a, b]$ is subdivided into a collection of subintervals $[a_i, b_i]$, such that $a_{i+1} = b_i$, and apply one of the schemes *EuExp*, *EuImp* or *EuComb* (with a reasonably small m) on each. Such an algorithm is very similar to a Runge-Kutta method; it is essentially a single-step algorithm with limited storage requirements, it is easy to implement adaptively, and its stability properties are not obivous *a priori*. It turns out that for non-stiff problems, *EuExp* works extremely well for a variety of combinations of the parameters m, J. For stiff problems, schemes of the *EuExp* type are obviously useless, since they are driven by the explicit Euler method. Schemes based on *EuImp* result in acceptable methods for certain choices of m, J. Unfortunately, for larger values of m, J, the stability properties of $EuImp_m^J$ deteriorate rapidly. Finally, schemes based on $EuComb$ result in acceptable stability properties for many values of m_1, j_1, m_2, j_2. All such methods are L-stable, and many combinations of m_1, m_2, j_1, j_2 result in nearly A-stable schemes (see section 5.3 below). While no general analysis of such methods has been carried out, our experiments appear to indicate that there exist schemes of this type that are of arbitrarily high order and are $A(\alpha)$-stable with α extremely close to 90°.

5 Stability and Accuracy Properties of Selected Schemes

We have implemented the schemes *EuExp*, *EuImp*, and *EuComb* in FORTRAN, and we have conducted a number of numerical experiments with the resulting codes in order to elucidate their performance. The following terminology is used in this section. The *stability region* associated with a numerical scheme for the solution of the equation (2.2) is defined to be the subset of the complex plane $\mathbb{C}$ consisting of all λ such that on the interval $[0, 1]$, the amplification factor define in (2.3) satisfies $Am(\lambda) \leq 1$. For a given $\epsilon > 0$, the *accuracy region* associated with a numerical scheme is defined to be the subset of $\mathbb{C}$ consisting of all λ such that, when the scheme is applied to the equation (2.2) on the interval $[0, 1]$,

$$| \widetilde{\varphi}(b) - \varphi(b) | < \epsilon. \tag{5.1}$$

Since both $\widetilde{\varphi}(b)$ and $\varphi(b)$ are analytic functions of λ, it follows from the maximum principle that both the stability and accuracy regions have well-defined boundaries. (For a more systematic approach to the analysis of stability regions, see [14]).

5.1 Stability and Accuracy Properties of EuExp schemes

We first compute the boundaries of the stability and accuracy regions for the schemes $EuExp_m^J$ for several choices of the parameters m and J (Figures $1-4$). It should be noted that the stability regions are compact; in other words, they are stable *inside* the boundaries marked $Am(\lambda) = 1$. Several observations can be made from these figures and from the more detailed numerical experiments we have performed.

1. In all cases, the stability condition is dominated by the accuracy condition. In other words, whenever $Re(\lambda) < 0$ and the scheme achieves a reasonable accuracy, the scheme is stable.

2. The sizes of both the stability and accuracy regions grow with the order of the scheme; when λ is purely imaginary, the scheme of order 20 requires about 20 nodes per wavelength to achieve 11 - digit precision.

REMARK 5.1. The Euler method is obviously not the most efficient solver to which the deferred correction approach can be applied. We have experimented, for example, with explicit Adams methods of orders up to 6, and have obtained improvements of up to a factor of three in terms of the number of function evaluations required.

5.2 Stability and Accuracy Properties of EuImp schemes

For a number of combinations m, J, we have numerically constructed the boundaries of the stability and accuracy regions for the schemes $EuImp_m^J$ (Figures $5-12$). The regions of stability of these schemes extend to infinity; in other words, they are stable *outside* the boundaries marked $Am(\lambda) = 1$. It is worth noting that, in most cases, the regions of *instability* are very much larger than the regions of accuracy. Thus, for each scheme we present two figures. The first is on a relatively coarse scale, depicting the stability region. The second is on a much finer scale, depicting the accuracy regions for two selected accuracies; in the latter case, the boundary of the stability region is virtually indistinguishable from the imaginary axis. Each of the figures carries a legend, specifying detailed stability characteristics of the scheme (all of the schemes $EuImp_m^J$ are $A(\alpha)$-stable, and the legends specify the approximate values of α, obtained numerically). None of the schemes $EuImp_m^J$ are L-stable; the legends specify the value of μ for each of the schemes (see (4.3)).

Several observations can be made from the figures 5 - 12.

1. There exist $EuImp_m^J$ schemes that are A-stable of order up to four (such as $EuImp_4^3$). The scheme $EuImp_6^5$ is $A(\alpha)$-stable with $\alpha > 89.5°$; for most practical purposes, such a scheme can be viewed as A-stable. For higher orders, the A - stability properties of $EuImp_m^J$ deteriorate rapidly (see Figures 9-12).

2. None of the schemes $EuImp_m^J$ are L-stable. However, in all cases we have studied, μ is less than 1/2; while L - stability (corresponding to $\mu = 0$) is very desirable, $\mu < 1/2$ guarantees a rate of decay that is sufficient in many cases.

3. The accuracy regions for all methods we have tested are very satisfactory, for both real and complex λ. It is easy to see, for example, from Figure 8 that $EuImp_6^5$ (a scheme of order 6) requires about 18 nodes per wavelength

to obtain 3 digits; the number increases to about 40 nodes per wavelength to obtain 5 digits, indicating the need for a higher order scheme. Such schemes are discussed in the following subsection.

5.3 Stability and Accuracy Properties for EuComb schemes

For a number of combinations m_1, m_2, j_1, j_2, we constructed numerically the boundaries of the stability and accuracy regions for the schemes $EuComb^{j_1,j_2}_{m_1,m_2}$ (Figures $13-24$). As in the case of the $EuComb$ schemes, the stability regions of these schemes extend to infinity; they are stable *outside* the boundaries marked $Am(\lambda) = 1$. As for the simpler $EuImp$ schemes, the regions of *instability* are generally very much larger than the regions of accuracy. Thus, we again present two figures for each case. The first is on a relatively coarse scale, depicting the stability region. The second is on a much finer scale, depicting the accuracy regions for two selected accuracies; in the latter case, the boundary of the stability region is virtually indistinguishable from the imaginary axis. Each of the figures carries a legend, specifying detailed stability characteristics of the scheme. All of the $EuComb^J_m$ schemes are $A(\alpha)$-stable, and the legends specify the approximate values of α, obtained numerically. Since all of the schemes $EuComb^J_m$ are L-stable (see Theorem 4.3), we do not specify the value of μ for each of the schemes.

Several observations can be made from Figures $13 - 24$.

1. There exist A-stable $EuImp^J_m$ schemes of order up to five (e.g., $EuComb^{5,5}_{6,5}$). For all orders we have tested (up to 30 or so), there exist $A(\alpha)$-stable schemes with α very close to $90°$. $EuComb^{12,12}_{13,12}$, for example, has order 12 (see Theorem 4.1), and is $A(\alpha)$-stable with $\alpha > 89.99°$. $EuComb^{19,19}_{20,19}$ has order 19 and is $A(\alpha)$-stable with $\alpha > 89.996°$. On the other hand, we are not able to predict reliably which of the schemes will have good stability properties, and which will not, until such properties are established numerically. $EuComb^{15,15}_{16,15}$, for example, has $\alpha > 89.99°$, while $EuComb^{16,16}_{17,16}$ has $89.01° < \alpha < 89.02°$. At the other extreme is the scheme $EuComb^{6,6}_{8,6}$, with a disastrous $\alpha < 56°$ (Figure 21). As a general rule, we have observed that the schemes $EuComb^{j_1,j_2}_{m_1,m_2}$ tend to have poor stability properties whenever $m_1 + m_2$ is even. This is, of course, mostly a curiosity, since good schemes of various orders are easily available.

2. The accuracy regions for all methods we tested are very satisfactory, for both real and complex λ. It is easy to see from Figure 24, for example, that the nineteenth order scheme $EuComb^{19,19}_{20,19}$ requires about 18 nodes per wavelength to obtain 10 digits.

6 Adaptive Implementation

In practical applications involving the numerical solution of ODEs, issues such as adaptive marching, accuracy control, etc. play an important role. Since the schemes of this paper are essentially of the single-step variety, most issues arising in their adaptive implementation are relatively simple. It is also worth keeping in mind that accuracy control is much simpler when the underlying solver has high convergence order.

We have implemented fully adaptive versions of the schemes *EuExp*, *EuImp*, *EuComb*. In this section, we describe some of the technical details of the implementation, while the following section describes some of the numerical experiments we have performed. We will be discussing these issues using *EuExp* as our model; the other two schemes (*EuImp* and *EuComb*) encounter identical problems, which are handled in a similar manner.

6.1 Accuracy control

Given an initial value problem (1.1), (1.2) on the interval $[a, b]$, an approximate solution
$EuExp_m^J(F, \phi_a)$, and a positive ϵ, we would like to determine whether

$$| \; EuExp_m^J(F, \varphi_a) - \varphi \; | < \epsilon. \tag{6.1}$$

Obviously, this cannot be done with complete reliability; the purpose of all existing techniques is to make the determination with very high probability, at an acceptable cost. Fortunately, the internal structure of the method $EuExp_m^J$ provides us with a number of conditions which can be checked. Taken together, the conditions listed below have been completely reliable in our experience.

1. We verify that the correction process has converged to the precision ϵ. In other words, we require that the norm of the vector δ (see (2.19)), obtained during the last correction, be less than ϵ. This does not guarantee (6.1); it does indicate that the correction scheme is internally consistent to precision ϵ.

2. The approximate solution $EuExp_m^J(F, \varphi_a)$ is obtained at Gaussian nodes on the interval $[a, b]$. We apply the operator W^m to $EuExp_m^J(F, \varphi_a)$, obtaining the m coefficients of its Legendre expansion (see (2.31)). If the discretization of $EuExp_m^J(F, \varphi_a)$ is sufficiently fine, the last several coefficients of the Legendre expansion must be small. In our implementation, we demand that the last two coefficients be smaller than ϵ.

3. Yet another test we perform attempts to verify simultaneously that both the correction process and the discretization have converged to precision ϵ. Once $EuExp_m^J(F, \varphi_a)$ has been evaluated, we obtain the value of the approximate solution at the point b via interpolation (see Observation 4.1). We apply the interpolation process to both $EuExp_m^J(F, \varphi_a)$ and $EuExp_m^{J-1}(F, \varphi_a)$, and demand that the difference be less that ϵ.

4. In extreme cases, the solution of the ODE can be so underresolved as to become unstable; the usual result is exponential overflow. To guard against this condition, we check the size of the solution at every step of the Euler process; if the solution is sufficiently large (we have arbitrarily

set the threshold to 10^{35}) the point of view is taken that the problem is underresolved.

6.2 Step-length control

Our approach here is completely standard. We start with a more or less arbitrary step-size, and attempt to apply the scheme $EuExp_m^J$. If the resulting precision is insufficient (according to the criteria (a) - (d) above), the step-length is halved, and the process repeated. If the precision is satisfactory, the steplength is unchanged. If the precision is satisfactory two steps in a row, the step-length is doubled.

6.3 Linearly implicit implementation

In order to reduce the number of function evaluations, a common practice in most extrapolation codes is to use a "linearly implicit" formulation of the marching scheme [11]. For the ordinary differential equation

$$\varphi'(t) = F(t, \varphi(t)) \qquad t \in [a, b] \tag{6.2}$$

with an approximate solution $\varphi_0(t)$, we let $\varphi = \varphi_0 + \delta$ and write (6.2) in the form

$$\varphi_0'(t) + \delta'(t) = F(t, \varphi_0(t) + \delta(t))$$

or

$$\delta(t) = \int_a^t F(\tau, \varphi_0(\tau) + \delta(\tau))d\tau - \varphi_0(t). \tag{6.3}$$

But for small δ, we have

$$F(t, \varphi_0(t) + \delta(t)) = F(t, \varphi_0(t)) + J_{\varphi_0}(t)\,\delta(t) + O(\|\delta\|^2),$$

where $J_{\varphi_0}(t)$ denotes the Jacobian of F with respect to its second argument at the point $(t, \varphi_0(t))$. Substituting this expression into (6.3), we get

$$\delta(t) = \int_a^t J_{\varphi_0}(t)\,\delta(\tau))d\tau + \int_a^t F(\tau, \varphi_0(\tau))d\tau - \varphi_0(t). \tag{6.4}$$

We can then use the backward Euler method to drive a deferred correction process applied to (6.4). Our linearly implict code performs up to six steps of deferred correction on this equation, after which the approximate solution φ_0 is updated according to

$$\varphi_0(t) := \varphi_0(t) + \delta(t).$$

Table 7.1: Performance of low, moderate, and high order spectral deferred correction methods for our first (non-stiff) example. The first column indicates the requested precision and the remaining columns list the number of function calls required by the corresponding scheme.

Precision	$EuExp_4^3$	$EuExp_6^5$	$EuExp_{16}^{15}$
10^{-3}	70	44	93
10^{-6}	287	176	155
10^{-12}	—	2574	310

7 Numerical Experiments

We illustrate the performance of spectral deferred correction methods with two examples. The first is the system of three ordinary differential equations satisfied by the Jacobi elliptic functions sn, cn, dn:

$$\begin{aligned} sn'(t) &= cn(t) \cdot dn(t) \\ cn'(t) &= -sn(t) \cdot dn(t) \\ dn'(t) &= -\mu \cdot sn(t) \cdot cn(t) \end{aligned}$$

with $\mu = 0.5$ on the interval $[0, 1]$ with initial data $sn(0) = 0$, $cn(0) = 1$, $dn(0) = 1$. This is a common model for nonstiff problems. As can be seen from the results in Table 1, the order of accuracy of the method should increase with the desired precision to obtain optimal performance.

Our second example is the Van der Pol oscillator, a well-known stiff system of two equations:

$$\begin{aligned} y_1'(t) &= y_2(t) \\ y_2'(t) &= (1 - y_1^2(t)\, y_2(t))/\epsilon - y_1(t) \end{aligned}$$

with $y_1(0) = 2$, $y_2(0) = 0$. We choose $\epsilon = 10^{-6}$ and solve on the interval $[0, 1]$. For the sake of comparison, we tested our codes against one of the best-performing codes for this problem - the high order extrapolation code EULSIM [4]. Our results are collected in Tables 2-4.

A number of observations can be made from these tables.

1. The code EULSIM requires noticeable fewer function evaluations than the *EuImp* schemes at any requested precision. EULSIM also requires fewer function evaluations than the linearly implicit deferred correction code.

2. If actual accuracies are compared, rather than requested precision, a slightly different picture begins to emerge. The singly implicit deferred correction scheme achieves about eight digits of accuracy at a requested tolerance of 10^{-5}, using 4,839 function calls. EULSIM, on the other hand, achieves

Table 7.2: Performance of the extrapolation code EULSIM on the Van der Pol oscillator problem. The first column indicates the requested precision, the second column indicates the number of function evaluations, the third column lists the computed solution component $y_1(2)$ and the fourth column lists the computed solution component $y_2(2)$.

Precision	F. calls	$y_1(2)$	$y_2(2)$
10^{-1}	166	-1.79765618619	0.000017424290
10^{-2}	250	-2.00019319529	0.000060543984
10^{-3}	751	1.98059515302	-0.033130626096
10^{-4}	1197	1.70727095158	-0.885933089211
10^{-5}	1872	1.70633329346	-0.892323342916
10^{-6}	2622	1.70618445916	-0.892674570523
10^{-7}	3870	1.70616982299	-0.892804713111
10^{-8}	5476	1.70616796672	-0.892809443466
10^{-9}	6531	1.70616775897	-0.892809533238
10^{-10}	10490	1.70616773572	-0.892809663387
10^{-11}	21547	1.70616773312	-0.892809699177
10^{-12}	86899	1.70616773227	-0.892809700614

Table 7.3: Performance of the spectral deferred correction code *EuImp* on the Van der Pol oscillator problem. The first four columns correspond to those in Table 2. The last two columns show the number of points n used on each subinterval (the maximal order of accuracy) and the maximal number of corrections *ncorr* actually used by the code.

Precision	F. calls	$y_1(2)$	$y_2(2)$	n	$ncorr$
10^{-1}	3462	1.70645342840	-0.892494581060	6	5
10^{-2}	5340	1.70642671028	-0.892530184336	8	4
10^{-3}	6754	1.70615323588	-0.892825220721	8	4
10^{-4}	11920	1.70616745993	-0.892809992650	16	8
10^{-5}	17880	1.70616771499	-0.892809719348	22	12
10^{-6}	20576	1.70616773089	-0.892809702550	22	12
10^{-7}	21852	1.70616773187	-0.892809701498	22	12
10^{-8}	23366	1.70616773221	-0.892809701142	22	12
10^{-9}	29798	1.70616773217	-0.892809701012	22	12
10^{-10}	56562	1.70616773217	-0.892809701047	22	12
10^{-11}	128362	1.70616773217	-0.892809701031	22	12

Table 7.4: Performance of the linearly implicit spectral deferred correction code on the Van der Pol oscillator problem. The columns correspond to those in Table 3.

Precision	F. calls	$y_1(2)$	$y_2(2)$	n	$ncorr$
10^{-1}	1976	1.70756400695	-0.891306260408	6	5
10^{-2}	2587	1.70629621863	-0.892672011388	8	4
10^{-3}	3112	1.70618777664	-0.892788115757	8	4
10^{-4}	4203	1.70616787726	-0.892809527001	16	8
10^{-5}	4839	1.70616773785	-0.892809694425	22	12
10^{-6}	5644	1.70616773478	-0.892809697543	22	12
10^{-7}	5887	1.70616773216	-0.892809699781	22	12
10^{-8}	7817	1.70616773207	-0.892809701150	22	12
10^{-9}	13885	1.70616773217	-0.892809700662	22	12
10^{-10}	28857	1.70616773217	-0.892809700900	22	12
10^{-11}	54528	1.70616773217	-0.892809701003	30	20

eight digits of accuracy at a requested tolerance of 10^{-10}, using 10,490 function calls. This is not intended to disparage EULSIM's performance. Our implementation simply has very strict (perhaps excessive) error control. If CPU times were compared, EULSIM would be the clear winner. Another interesting comparison can be made at ten digits of accuracy. The singly implicit deferred correction scheme requires 5,887 function calls while the RADAU code of Hairer and Wanner [11], which is more efficient in this regime than EULSIM, requires 6,517.

8 Conclusions

We believe that deferred correction methods based on an integral equation formulation of the ordinary differential equation are promising candidates for further investigation. They have excellent stability properties, are easy to implement, and require only a good low-order solver to drive the process. Our preliminary experiments, using a primitive adaptive implementation, compare favorably with a state-of-the-art extrapolation code at moderate to high precision.

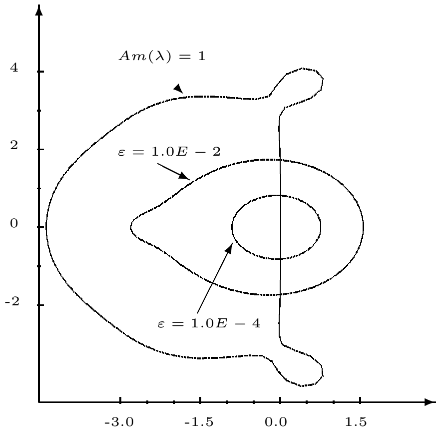

Figure 1:
Stability and accuracy regions for
$EuExp_4^3$

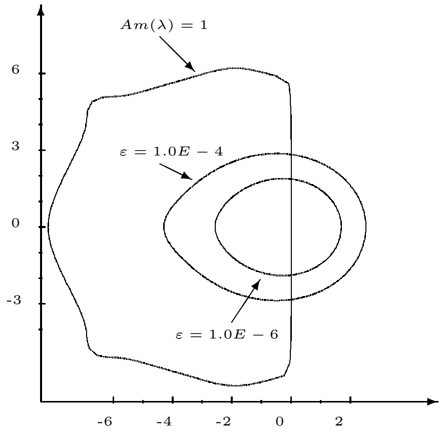

Figure 2:
Stability and accuracy regions for
$EuExp_8^7$

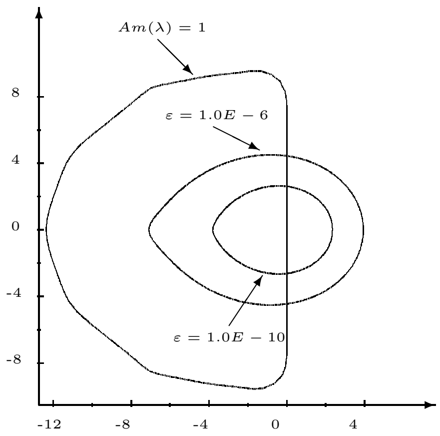

Figure 3:
Stability and accuracy regions for
$EuExp_{13}^{12}$,

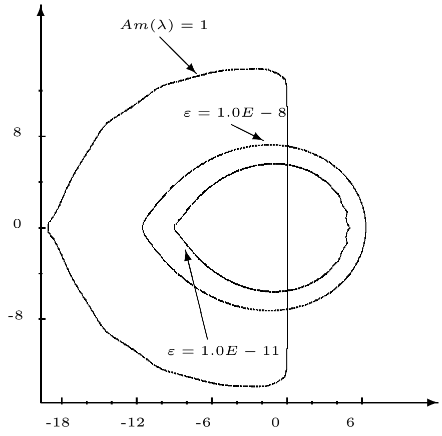

Figure 4:
Stability and accuracy regions for
$EuExp_{20}^{19}$

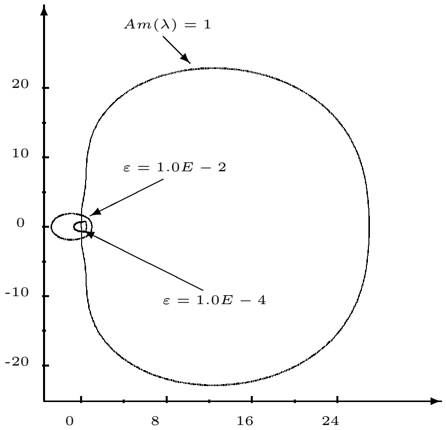

Figure 5:
Stability and accuracy regions for $EuImp_4^3$; $\mu \approx -.3913$, $\alpha = 90°$

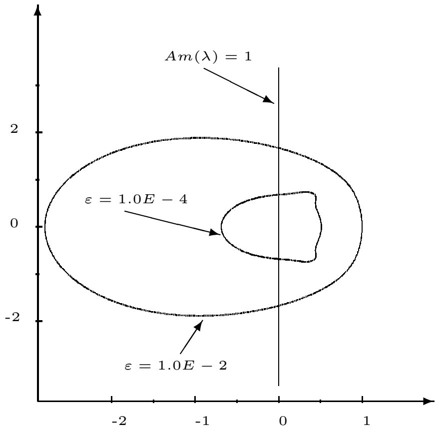

Figure 6:
Detail of stability and accuracy regions for $EuImp_4^3$

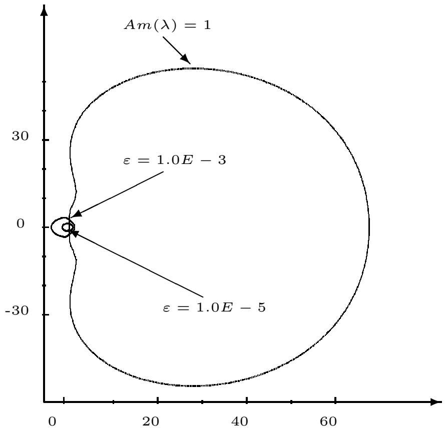

Figure 7:
Stability and accuracy regions for $EuImp_6^5$; $\mu \approx -.3101$, $\alpha \approx 89.979°$

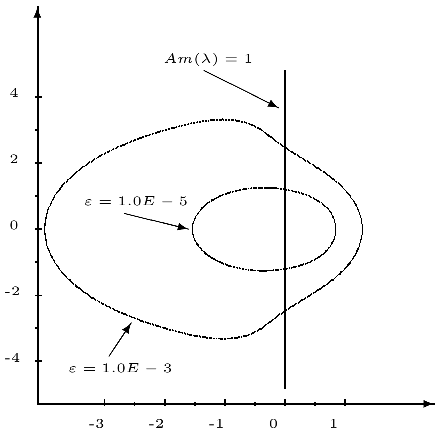

Figure 8:
Detail of stability and accuracy regions for $EuImp_6^5$

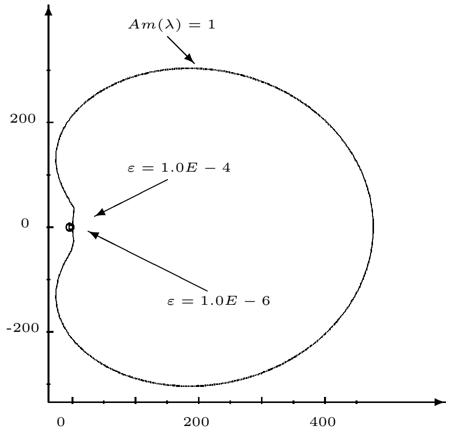

Figure 9:
Stability and accuracy regions for $EuImp_{12}^{11}$; $\mu \approx 0.1369$, $\alpha \approx 76.8°$

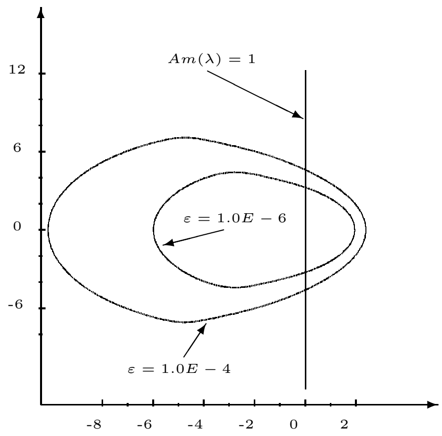

Figure 10:
Detail of stability and accuracy regions for $EuImp_{12}^{11}$

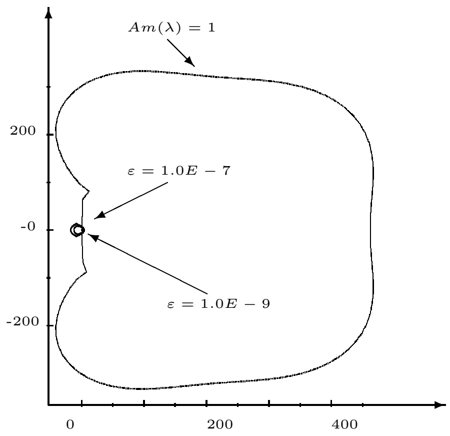

Figure 11:
Stability and accuracy regions for $EuImp_{20}^{19}$; $\mu \approx -0.3030$, $\alpha \approx 77.5°$

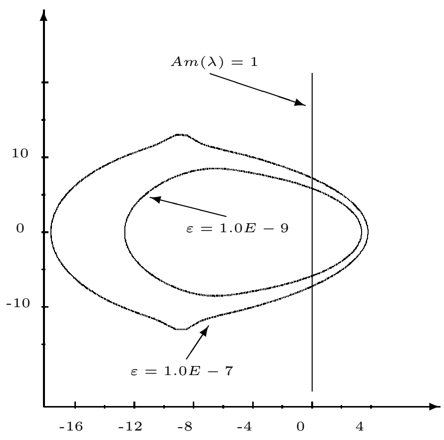

Figure 12:
Detail of stability and accuracy regions for $EuImp_{20}^{19}$

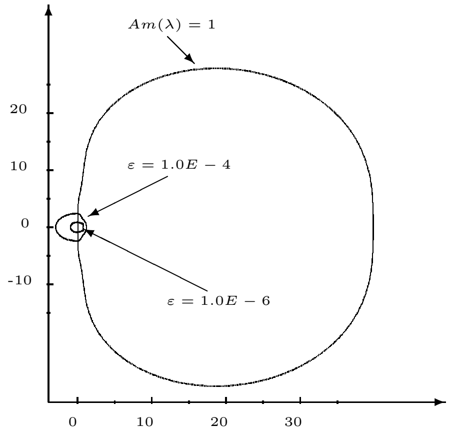

Figure 13:
Stability and accuracy regions for $EuComb^{5,5}_{6,5}$; $\alpha = 90°$

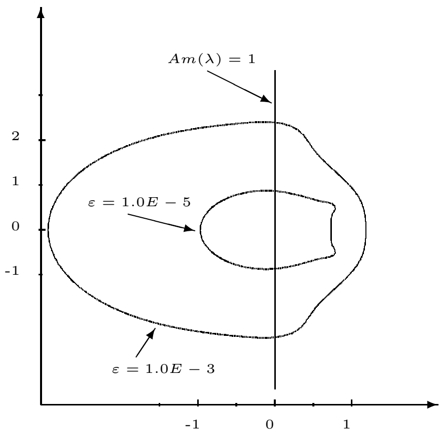

Figure 14:
Detail of stability and accuracy regions for $EuComb^{5,5}_{6,5}$

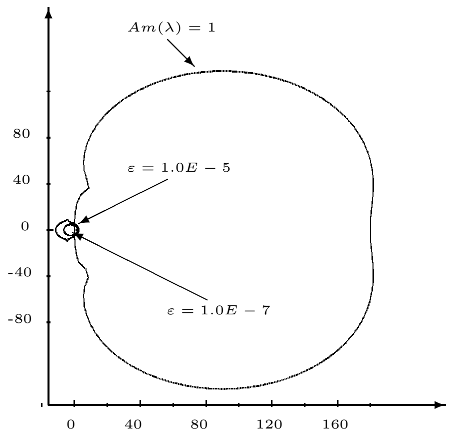

Figure 15:
Stability and accuracy regions for $EuComb^{12,12}_{13,12}$; $\alpha \approx 89.9914°$

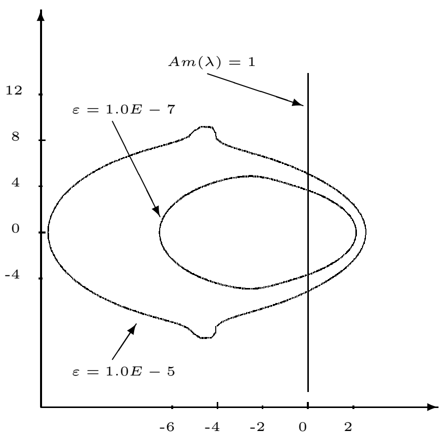

Figure 16:
Detail of stability and accuracy regions for $EuComb^{12,12}_{13,12}$

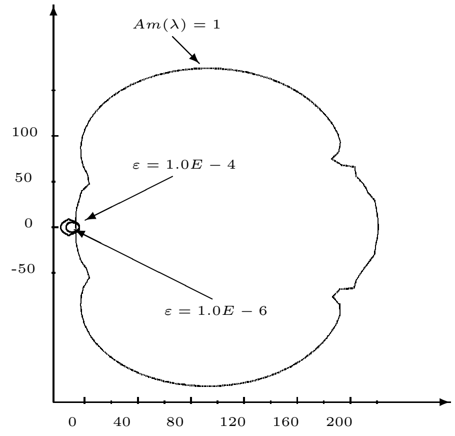

Figure 17:
Stability and accuracy regions for
$EuComb^{15,15}_{16,15}$; $\alpha \approx 89.994°$

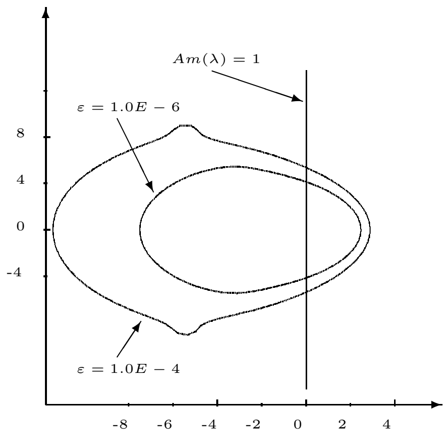

Figure 18:
Detail of stability and accuracy
regions for $EuComb^{15,15}_{16,15}$

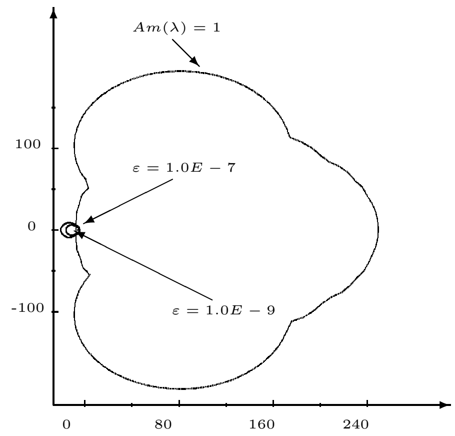

Figure 19:
Stability and accuracy regions for
$EuComb^{16,16}_{17,16}$; $\alpha \approx 89.014°$

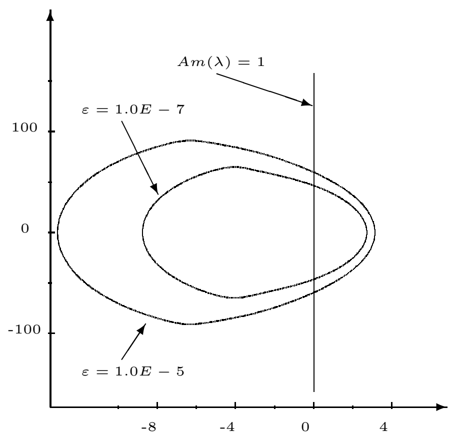

Figure 20:
Detail of stability and accuracy
regions for $EuComb^{16,16}_{17,16}$

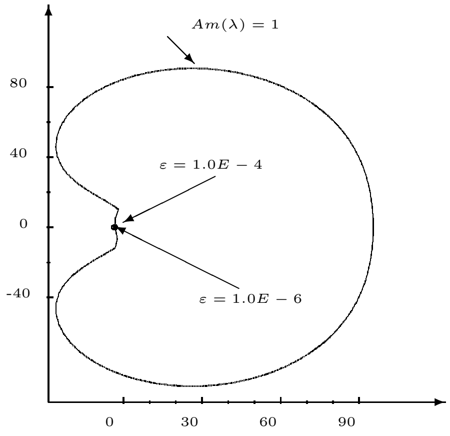

Figure 21:
Stability and accuracy regions for
$EuComb^{6,6}_{8,6}$; $\alpha \approx 55.786°$

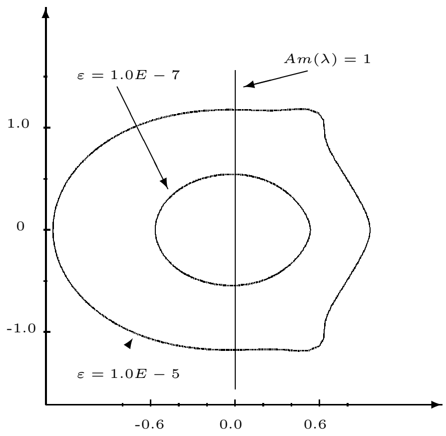

Figure 22:
Detail of stability and accuracy
regions for $EuComb^{6,6}_{8,6}$

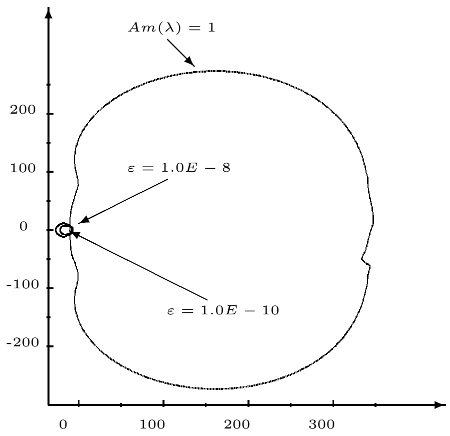

Figure 23:
Stability and accuracy regions for
$EuComb^{19,19}_{20,19}$; $\alpha \approx 89.9969°$

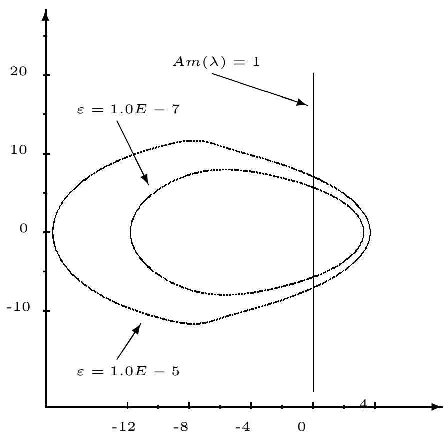

Figure 24:
Detail of stability and accuracy
regions for $EuComb^{19,19}_{20,19}$

REFERENCES

1. K.E. Brenan, S.L. Campbell, and L.R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1995.
2. J. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, Wiley, 1987.
3. K. Böhmer, H.J. Stetter, eds., *Defect Correction Methods, Theory and Applications*, Springer-Verlag, Wien-New York, 1984.
4. P. Deuflhard: *Recent Progress in Extrapolation Methods for Ordinary Differential Equations*, SIAM Rev. **27**, 505–535 (1985).
5. A. Dutt, M. Gu, V. Rokhlin, *Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation* SIAM J. Num. Anal. **33**, 1689–1711 (1996).
6. R. Frank and C. Ueberhuber, *Iterated Deferrred Correction for the Efficient Solution of Stiff Systems of Ordinary Differential Equations*, BIT **17** (1977), 146-159.
7. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, New Jersey, 1971.
8. D. Gottlieb, S.A. Orszag, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia, 1977.
9. L. Greengard, *Spectral Integration and Two-Point Boundary Value Problems*, SIAM J. Num. Anal. **28**, 1071-1080 (1991).
10. E. Hairer, S. P. Norsett and G. Wanner, *Solving Ordinary Differential Equations I, Non-Stiff Problems*, Springer Verlag, Berlin, 1993.
11. E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II*, Springer, 1996.
12. D. J. Higham and L. N. Trefethen, *Stiffness of ODEs*, BIT, **33**, 285–303 (1993).
13. Arieh Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, 1996.
14. R. Jeltsch and O. Nevanlinna, *Stability and Accuracy of Time Discretizations for Initial Value Problems*, Numer. Math. **40**, 245–296 (1982).
15. J. D. Lambert, *Numerical Methods for Ordinary Differential Equations*, Wiley, 1991.
16. B. Lindberg, *Error Estimation and Iterative Improvement for Discretization Algorithms*, BIT, **20**, 486–500 (1980).
17. V. Pereyra, *Iterated Deferred Correction for Nonlinear Boundary Value Problems*, Numer. Math. **11**, 111–125 (1968).
18. R. D. Skeel, *A Theoretical Framework for Proving Accuracy Results for Deferred Correction*, SIAM J. Numer. Anal. **19**, 171–196 (1981).
19. J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis* Springer, 1992.
20. L. N. Trefethen and M. R. Trummer, *An Instability Phenomenon in Spectral Methods*, SIAM J. Numer. Anal. **24**, 1008 - 1023 (1987).
21. P. E. Zadunaisky, *On the Estimation of Errors Propagated in the Numerical Integration of Ordinary Differential Equations*, Numer. Math. **27**, 21–40 (1976).