Spectral Deferred Correction for Non-stiff ODEs

Numerical Methods II Project Report

Amirhossein Khalilian-Gourtani
email: akg404@nyu.edu
N-number: N14959977

May 11, 2021

Abstract

In this project we investigate the spectral deferred correction method for ordinary differential equations (ODEs). We first review some preliminaries including the problem formulation, Picard integral equation, and spectral integration. Then, we review the spectral deferred correction method for non-stiff ODEs and implement a solver based on this method. In our implementation we use Chebyshev nodes for spectral integration and deferred correction. Additionally, we investigate the effect of an adaptive step-length control scheme. Finally, we show the performance of the implemented method on some example problems for both non-adaptive and adaptive step-size schemes.

Contents

1 Introduction

Solving initial value problems governed by systems of ordinary differential equations (ODEs) via numerical discretization is a well-studied subject. The goal here is to build efficient, stable, and accurate solvers for such problems. In lectures and the book we mostly covered a class of methods that consist of using high-order discretization schemes to achieve high accuracy such as Runge-Kutta methods. Note that for non-stiff problems there exists effective high-order discretizations based on such methods but for very accurate schemes the stability constraints become too restrictive. Another class of methods try to make a low-order method converge more rapidly by use of Richardson extrapolation. The downside of these methods is that they are computationally expensive as they require computing the solution on a refined grid.

In this project we are interested in a class of methods that try to accelerate the convergence of low-order schemes by utilizing deferred correction. More specifically, we review an approach to deferred correction that is based on solving the equivalent Picard integral equation on a composite Chebyshev grid. This method is known as spectral deferred correction [DGR00]. We implement this method for non-stiff ODEs and show numerical results for both non-adaptive and adaptive step-size cases. Note that the implementation in [DGR00] uses Gauss-Legendre grid points for spectral integration but it is discussed that the method is similarly effective when using Chebyshev nodes. For simplicity of implementation and in order to use the spectral integration that we developed in class we choose the Chebyshev grid in our implementation.

1.1 Notation and organization

We consider the initial value problem in the standard form as

$$u'(t) = F(t, u(t)), \quad t \in [a, b], \quad u(a) = u_a, \tag{1}$$

where $u(t) \in \mathbf{R}^n$ and $F : \mathbf{R} \times \mathbf{R}^n \to \mathbf{R}^n$. Following [DGR00], we assume that F is sufficiently smooth and for introducing the algorithm we assume that $n = 1$.

The rest of this report is organized as follows. In Section 2 we review Picard integral form of (1) and spectral integration on interval $[a, b]$ using Chebyshev nodes. In Section 3, we review the spectral deferred correction method introduced in [DGR00] and show our implementation with and without adaptive step-size. In Section 4, we show the order of accuracy achieved by our implementation. We also show the effect of adaptive step-size with numerical examples. Finally, Section 5 concludes the report. A list of the files and codes attached to this report and parts of important functions are presented in Section 6.

2 Preliminaries

2.1 Picard integral equation

If we integrate equation (1), with respect to t, we get the following Picard equation

$$u(t) = u_a + \int_a^t F(\tau, u(\tau))d\tau. \tag{2}$$

Suppose that $t_0 < t_1 < \cdots < t_m < t_{m+1}$ is a refinement of the interval $[a, b]$ with $t_0 = a$ and $t_{m+1} = b$. Then the forward Euler method for (2) has the following steps

$$u_{i+1} = u_i + k_i\, F(t_i, u_i), \quad k_i = t_{i+1} - t_i, \quad \text{with} \quad i = 0, 1, \cdots, m. \tag{3}$$

2.2 Spectral integration

Let $r_i = \cos(\frac{i\pi}{m})$ for $i = 0, 1, \cdots, m$ denote the practical Chebyshev nodes on interval $[-1, 1]$. As we discussed in our first homework, once we have the approximation of the form

$$F(t) = \sum_{i=0}^{m} \alpha_i T_i(t) \tag{4}$$

where $T_i(t)$ is the ith Chebyshev polynomial it is straightforward to find an expression for the integral of $F(t)$ using the fact that

$$\int T_i(t)dt = \frac{1}{2}\left(\frac{T_{i+1}(t)}{i+1} - \frac{T_{i-1}(t)}{i-1}\right) \quad \text{for} \quad i \geq 2. \tag{5}$$

Our goal is to find the coefficients β_i such that $\int_{-1}^{t} F(\tau)d\tau = \sum_{i=0}^{m} \beta_i T_i(t)$. To find the coefficients β_i in term of α_i we have

$$\int_{-1}^{t} F(\tau)d\tau = \int_{-1}^{t} \sum_{i=0}^{m} \alpha_i T_i(\tau)d\tau = \sum_{i=0}^{m} \alpha_i \int_{-1}^{t} T_i(\tau)d\tau.$$

Note that for $i \geq 2$ we have

$$\int_{-1}^{t} T_i(\tau)d\tau = \frac{1}{2}\left(\frac{T_{i+1}(\tau)}{i+1} - \frac{T_{i-1}(\tau)}{i-1}\right)\Bigg|_{\tau=-1}^{\tau=t}.$$

As a result we have

$$\begin{aligned}
&\int_{-1}^{t} F(\tau)d\tau = \sum_{i=0}^{m} \alpha_i \int_{-1}^{t} T_i(\tau)d\tau = \alpha_0 \int_{-1}^{t} T_0(\tau)d\tau + \alpha_1 \int_{-1}^{t} T_1(\tau)d\tau + \sum_{i=2}^{m} \alpha_i \int_{-1}^{t} T_i(\tau)d\tau \\
&= \alpha_0 \int_{-1}^{t} 1 d\tau + \alpha_1 \int_{-1}^{t} \tau d\tau + \sum_{i=2}^{m} \frac{\alpha_i}{2}\left(\frac{T_{i+1}(\tau)}{i+1} - \frac{T_{i-1}(\tau)}{i-1}\right)\Bigg|_{\tau=-1}^{\tau=t} \\
&= \alpha_0(t+1) + \frac{\alpha_1}{2}(t^2 - 1) + \sum_{i=2}^{m} \frac{\alpha_i}{2}\left(\frac{T_{i+1}(t)}{i+1} - \frac{T_{i-1}(t)}{i-1} - \frac{T_{i+1}(-1)}{i+1} + \frac{T_{i-1}(-1)}{i-1}\right)
\end{aligned}$$

Note that $T_0(t) = 1,\ T_1(t) = t,\ t^2 = \frac{1}{2}(T_2(t) + T_0(t))$. As a result, we can set β_i as the coefficient of the term $T_i(t)$ in the above equation. Consequently, we get

$$\begin{aligned}
\beta_0 &= \alpha_0 - \frac{\alpha_1}{4} + \sum_{i=2}^{m} \frac{\alpha_i}{2}\left(\frac{T_{i-1}(-1)}{i-1} - \frac{T_{i+1}(-1)}{i+1}\right), \\
\beta_1 &= \alpha_0 - \frac{\alpha_2}{2}, \\
\beta_i &= \frac{\alpha_{i-1} - \alpha_{i+1}}{2i} \quad i \geq 2
\end{aligned} \tag{6}$$

where, $T_i(-1) = (-1)^i$, for $i \geq 0$. Given the coefficients β_i we have

$$\int_{-1}^{t} F(\tau)d\tau = \sum_{i=0}^{m} \beta_i T_i(t). \tag{7}$$

For a more general interval $[a, b]$ consider the nodes $s_0, s_1, \cdots, s_m$ given by

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

Then by a change of variables we can write

$$\int_{a}^{s} F(\tau)d\tau = \frac{b-a}{2} \sum_{i=0}^{m} \beta_i T_i\left(\frac{2}{b-a}\left(s - \frac{b+a}{2}\right)\right). \tag{8}$$

In our implementation, the function `ChebFFT` uses fast Fourier transform (FFT) algorithm to compute the coefficients α_i given a vector of function values on interval $[-1, 1]$. The function `iChebFFT` computes the function values given Chebyshev coefficients. The function `ChebInt` uses these functions to calculate the integral of the form (7) on interval $[-1, 1]$. For integrals of the form (8) we can scale the output of `ChebInt` by the factor $\frac{b-a}{2}$ for interval $[a, b]$.

In cases that we have ODE systems with $n > 1$, we have $F(t) \in \mathbf{R}^n$ and we need to calculate the integral for each component of F separately. If we consider the function values in a matrix where time-steps are arranged in each column, we can apply the `fft` function row-wise and compute the integral for each component. Our implementation of `ChebFFT, iChebFFT,` and `ChebInt` uses this property for systems of equations.

3 Spectral deferred correction method

3.1 Spectral deferred correction algorithm

Here we review the spectral deferred correction (SDC) scheme as presented in [DGR00]. Consider the ODE of the form

$$u'(t) = F(t, u(t)), \quad t \in [t_a, t_b], \quad u(t_a) = u_a. \tag{9}$$

Let us assume a fixed step-size k and divide the time interval $[t_a, t_b]$ into smaller subintervals of length k. For each subinterval $[a, b]$ we can use the Chebyshev nodes $s_i = \frac{b-a}{2} r_i + \frac{b+a}{2}$ for $i = 0, \cdots, m$ to apply the spectral deferred correction method. The division of the interval is shown pictorially in figure 1.

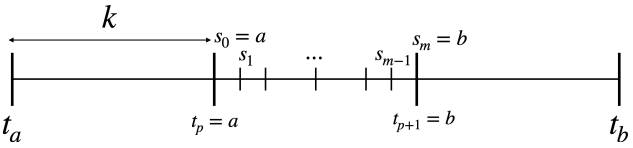


Figure 1: Pictorial depiction of how the interval $[t_a, t_b]$ is divided into subintervals of length k and for each subinterval Chebyshev nodes s_i are utilized.

The Picard integral equation in (2) for a subinterval $[a, b]$ has the form

$$\phi(s) = \phi_a + \int_a^s F(\tau, \phi(\tau)) d\tau.$$

Note that for SDC we use $\phi(s_i)$ instead of $u(t)$ to emphasize that the equation is on a subinterval and the time points correspond to Chebyshev nodes. Now assume that we have an initial approximate solution $\phi^0(s)$. The residual function can be written as

$$\epsilon(s) = \phi_a + \int_a^s F(\tau, \phi^0(\tau))\, d\tau - \phi^0(s). \tag{10}$$

Define the error $\delta(s)$ by $\delta(s) = \phi(s) - \phi^0(s)$. Additionally, define

$$G(s, \delta) = F(s, \phi^0(s) + \delta(s)) - F(s, \phi^0(s)). \tag{11}$$

Following the derivation in [DGR00], we get

$$\delta(s) - \int_a^s G(\tau, \delta(\tau))\, d\tau = \epsilon(s) \tag{12}$$

which is a Picard-type integral equation.

In order to solve (12) first we need to approximate $\epsilon(s)$ defined in (10). Given the nodes $s_0, \cdots, s_m$, let $\phi^j = (\phi_0^j, \cdots, \phi_m^j) \approx (\phi(s_0), \cdots, \phi(s_m))$ and $\overline{F}(\phi^j) = \big(F(s_0, \phi_0^j), \cdots, F(s_m, \phi_m^j)\big)$. The integral $\int_a^s F(\tau, \phi^j(\tau))$ in (10) can be approximated using the spectral integration as in equation (8), which we denote by $S_m(\overline{F})$. Then, the residual function $\epsilon(s)$ can be approximated by

$$\sigma^j = S_m(\overline{F}) - \phi^j + (\phi_a, \cdots, \phi_a).$$

Given values of σ^j, the equation (12) can be solved by a forward Euler iteration for δ_i as

$$\delta_{i+1}^{j+1} = \delta_i^{j+1} + (s_{i+1} - s_i) G(s_i, \delta_i^{j+1}) + \sigma_{i+1}^j - \sigma_i^j, \quad i = 0, \cdots, m-1.$$

Then the approximation can be updated as $\phi^{j+1} = \phi^j + \delta^{j+1}$.

We apply the correction for a total of J repetitions. Once the subinterval is resolved we move to the next subinterval. Note that since the Chebychev nodes include the end-points, we can initialize the next subinterval with the calculated value from the previous one. The algorithm is summarized in algorithm 1.

Algorithm 1: Spectral Deferred Correction with fixed step-size

Get: function $F(u(t), t)$, initial value u_a, and interval $[t_a, t_b]$;
Set: step-size k, number of Chebyshev points m, and number of corrections J;
Let r_i for $i = 0, \cdots, m$ be Chebyshev nodes on interval $[-1, 1]$;
`//Divide the interval` $[t_a, t_b]$ `with step-size` k
$t_p = t_a + p \times k \quad p = 0, \cdots, \text{round}(t_b/k)$;
$u(t_a) \leftarrow u_a$ `//set initial value`
for $p = 0, \cdots, \text{round}(t_b/k)$ **do**
  $a = t_a + p \times k$ and $b = t_a + p \times (k+1)$ `//subinterval` $[a, b]$
  $s_i = \frac{b-a}{2} r_i + \frac{b+a}{2}$;
  $\phi_0^0 \leftarrow u(a)$ and $\bar{\phi}_a = [\phi_0^0, \cdots, \phi_0^0]$;
  `//run forward Euler to get initial approximation of` $\phi^0(s_i)$
  for $i = 0, \cdots, m-1$ **do**
    $\phi_{i+1}^0 = \phi_i^0 + (s_{i+1} - s_i) \times F(s_i, \phi_i^0)$;
  `//run correction steps`
  for $j = 1, \cdots, J$ **do**
    `//compute approximate residual` $\sigma^{j-1} \approx \epsilon\left(\phi^{j-1}\right)$
    $S_m(\overline{F}) = \frac{b-a}{2} \text{ChebInt}\left(F\left(s, \phi^{j-1}\right)\right)$;
    $\sigma^{j-1} = S_m(\overline{F}) - \phi^{j-1} + \bar{\phi}_a$;
    `//compute the error` δ^j `with forward Euler`
    $\delta_0^j = \sigma_0^{j-1}$;
    for $i = 0, \cdots, m-1$ **do**
      $G(s_i, \delta_i^j) = F(s_i, \phi_i^{j-1} + \delta_i^j) - F(s_i, \phi_i^{j-1})$;
      $\delta_{i+1}^j = \delta_i^j + (s_{i+1} - s_i) \times G(s_i, \delta_i^j) + \sigma_{i+1}^{j-1} - \sigma_i^{j-1}$;
    `//update the approximate solution`
    $\phi^j = \phi^{j-1} + \delta^j$;
  $u(b) \leftarrow \phi_m^j$ `//set the solution at end of subinterval`
Output: $t,\ u(t)$

3.2 Adaptive step-size implementation

In the previous section we considered the case where step-size k is fixed. In a more practical setting, we can adaptively change the step-size based on the properties of the problem in hand and also the error tolerance. For a subinterval $[a, b]$ let ϕ^J denote the approximate solution after J steps of deferred correction. Then, for a given precision η, we want $|\phi^J - \phi_{\text{true}}| < \eta$. Following the accuracy control scheme in [DGR00], we check the following conditions to ensure the accuracy and change the step-size if required accordingly. The conditions are as follows:

1. We verify that the correction process has converged to the precision η. In other words, we require that $\|\delta^J\|_\infty \leq \eta$.

2. If the discretization is sufficiently fine, the last several coefficients of the Chebyshev

expansion of ϕ^J must be small. In our implementation, we demand that the last two coefficients be smaller than η.

3. In another case the solution of the ODE can become unstable as a result of large step-size. To avoid this, we check the size of the solution after forward Euler and require that $|\phi_i^0| < 10^{35}$ for each subinterval.

In order to adaptively change the step-size, we follow the discussion in [DGR00]. In the adaptive step-size case, we start with an arbitrary step-size and apply the scheme. If any of the above precision requirements are violated, we halve the step-size and try again. If all the precision requirements are satisfied for two consecutive subintervals, the step-size is then doubled.

4 Numerical experiments

In this section we illustrate the performance of the implemented method on two test problems. For our first example we consider a simple **third order initial value problem** from the homework

$$v'''(t) + v''(t) + 4v'(t) + 4v(t) = 4t^2 + 8t - 10, \quad v(0) = -3,\ v'(0) = -2,\ v''(0) = 2. \quad (13)$$

Note that the solution to this equation is $v(t) = -\sin(2t) + t^2 - 3$. To write the equation in standard form we have

$$u_1(t) = v(t), \quad u_2(t) = v'(t), \quad u_3(t) = v''(t)$$

and we get the system of equations,

$$\begin{cases} u_1'(t) &= u_2(t), \\ u_2'(t) &= u_3(t), \\ u_3'(t) &= -u_3(t) - 4u_2(t) - 4u_1(t) + 4t^2 + 8t - 10. \end{cases} \quad (14)$$

For the initial condition we have $u(0) = (-3, -2, 2)^T$. We consider interval $[0, 2]$.

For the second example, we use the system of three ordinary differential equations satisfied by the **Jacobi elliptic functions** $sn,\ cn,\ dn$:

$$\begin{cases} sn'(t) = cn(t) \cdot dn(t), \\ cn'(t) = -sn(t) \cdot dn(t), \\ dn'(t) = -\mu \cdot sn(t) \cdot cn(t) \end{cases} \quad (15)$$

with $\mu = 0.5$ on the interval $[0, 1]$ with initial data $u(0) = (0, 1, 1)^T$. This is the example of non-stiff problem used in [DGR00]. We calculate the error in this case against the function values given by `ellipj` function in MATLAB.

4.1 Numerical examples with fixed step-size

In this section we show results for applying the spectral deferred correction method with a fixed step-size to our test problems. The solution of third order IVP (14)with spectral deferred correction with $m = 5$ and $J = 4$ for different values of step-size k is shown in figure 3. Additionally, the error values for each case and the result of least squares fit for this problem are shown in table 1. As we can observe the solution is recovered on each subinterval on the Chebyshev nodes and as we decrease k, the subintervals get more refined. The least squares fit to the error for this case is shown in table 1.

step-size k	**error**	**ration**	**observed order**	**function evals**
0.50	1.19e-4	-	-	120
0.20	1.61e-6	73.79	4.69	300
0.10	8.96e-8	17.96	4.17	600
0.05	5.41e-9	16.54	4.04	1200
Least squares fit gives $E(k) = 0.002\,k^{4.33}$.				

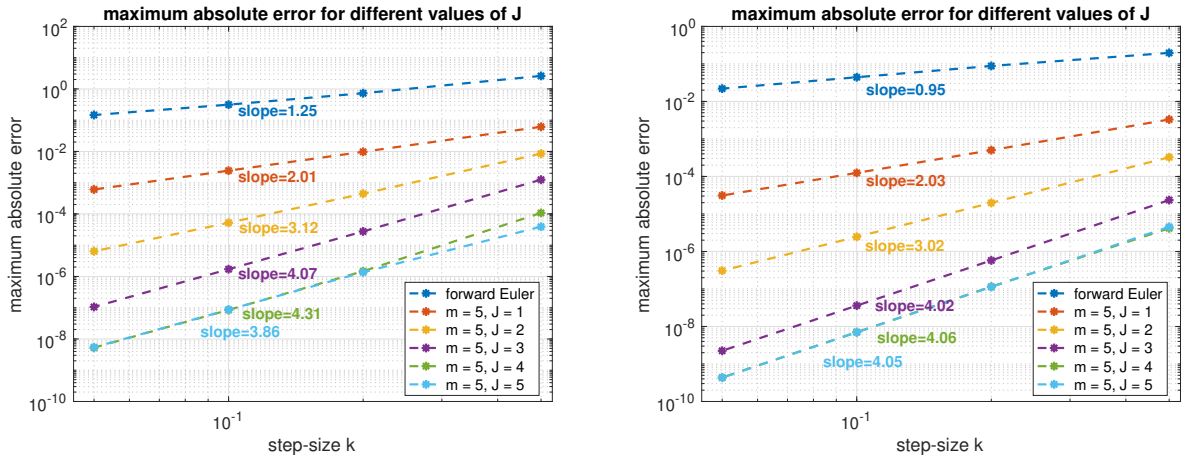
Table 1: Error values vs. step-size for third order IVP (14). In each case the step-size k is fixed and spectral deferred correction with $m = 5$ and $J = 4$ is used. The least squares fit to the error is also provided.

Similarly, the solution of Jacobi elliptic functions (15)with spectral deferred correction with $m = 5$ and $J = 4$ for different fixed step-sizes is shown in figure 4 and the corresponding error values are shown in table 2.

step-size k	**error**	**ration**	**observed order**	**function evals**
0.50	5.22e-6	-	-	60
0.20	1.23e-7	42.56	4.09	150
0.10	7.40e-9	16.58	4.05	300
0.05	4.47e-10	16.54	4.04	600
Least squares fit gives $E(k) = 8.6\text{e-}5\,k^{4.06}$.				

Table 2: Error values vs. step-size for Jacobi elliptic functions (15). In each case the step-size k is fixed and spectral deferred correction with $m = 5$ and $J = 4$ is used. The least squares fit to the error is also provided.

In figure 2 we show the loglog plot of error vs. step-size k for different values of J when $m = 5$ for both problems. In both cases a least squares fit is used to determine the slope of each line which shows the achieved empirical order of accuracy. We can observe that as J is increased not only the error gets smaller but also the slope of reduction is increased. As we can see the slope is close to $J + 1$ for most cases but when $J = m = 5$ and the number of correction steps is close to the number of nodes, there is diminishing returns from further correction steps in terms of order.



(a) Plot for third order IVP (14) **(b)** Plot for Jacobi elliptic functions (15)

Figure 2: Loglog plot of maximum absolute error vs. step-size k for forward Euler method and spectral deferred correction with $m = 5$ and different values of J. The least squares fit to slope is shown for each curve.

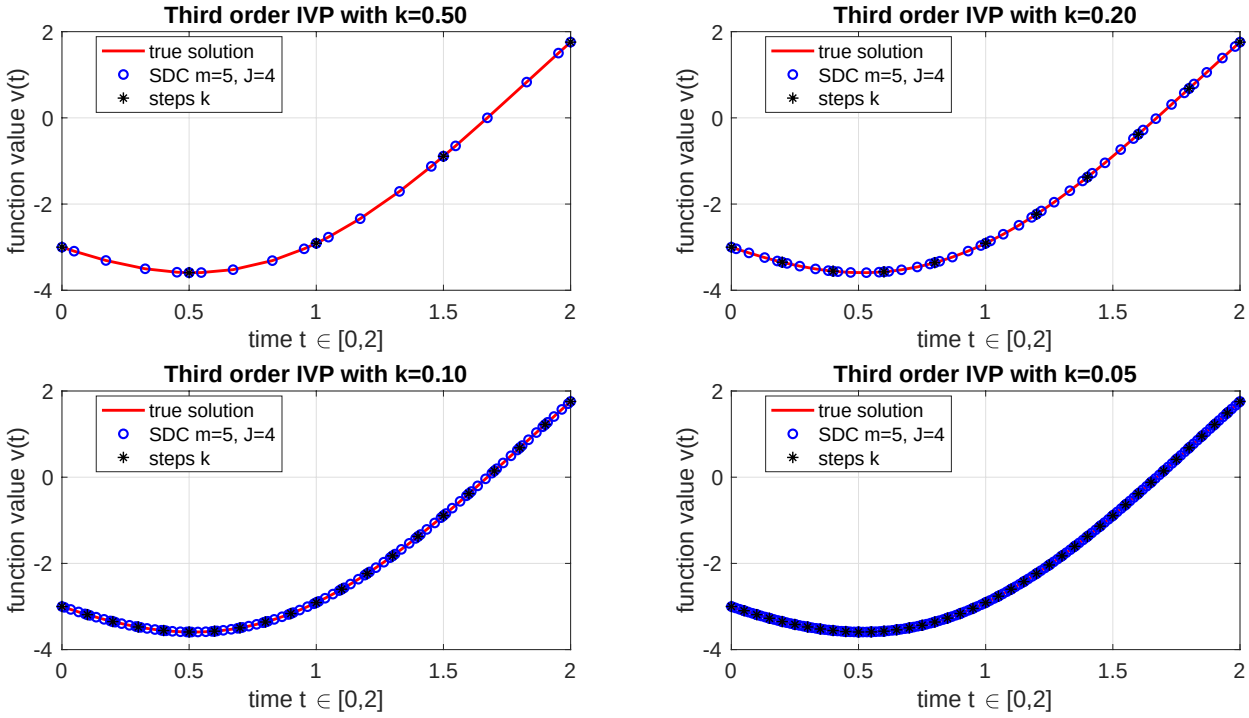


Figure 3: Solution of third order IVP (14) obtained by spectral deferred correction with $m = 5$, $J = 4$ for different values of step-size k. Blue circles show the computed solution on Chebyshev nodes in each subinterval. Black asterisks show the end-points of subintervals with step-size k. True solution $v(t)$ is shown with red solid line.

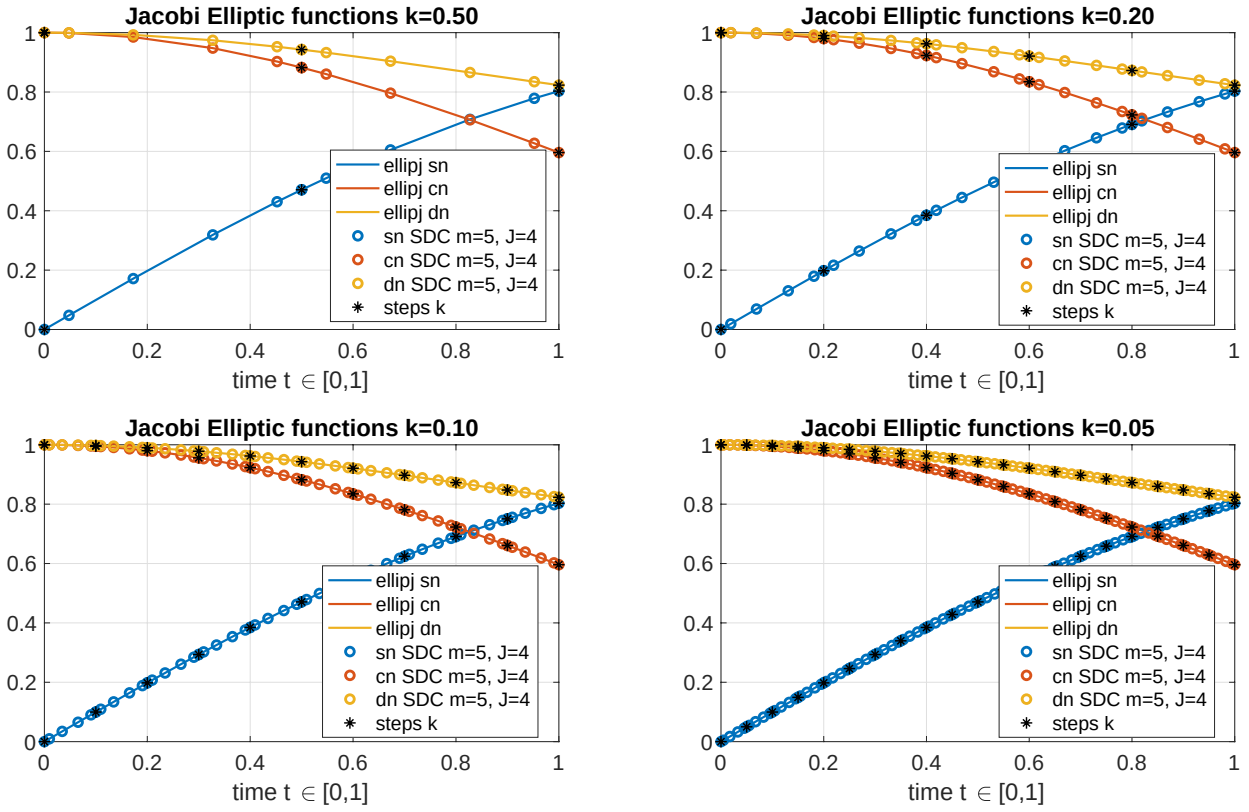


Figure 4: Solution of Jacobi elliptic functions (15) obtained by spectral deferred correction with $m = 5$, $J = 4$ for different values of step-size k. Circles show the computed solution on Chebyshev nodes in each subinterval. Black asterisks show the end-points of subintervals with step-size k. Reference solution $sn(t)$, $cn(t)$, $dn(t)$ from MATLAB ellipj are shown with solid lines.

4.2 Numerical examples with adaptive scheme

In this section we present the results for both of our test problems when the spectral deferred correction is utilized with adaptive step-size scheme. Figure 6 shows the solution of third order IVP (14)with $m = 5$ and $J = 4$ when step-size k is initially set to 0.2 for tolerances η =1e-3 and η =1e-6. Note that black asterisks show the end-points of subintervals. When tolerance is relatively large (1e-3), the algorithm increases the step-size gradually and takes larger steps (note that the first two intervals are size 0.2 and then the step is doubled). When a smaller tolerance is selected (1e-6), the adaptive step-size is reduced from the initial 0.2 (for example see the first few subintervals). Number of function evaluations given a requested precision for different values of m and J, are also shown in table 3. We can observe that a requested precision can be achieved with moderate number of function evaluations.

Precision	$m = 3,\ J = 2$	$m = 5,\ J = 4$	$m = 7,\ J = 6$
1e-3	1488	210	280
1e-6	46188	1620	392
1e-12	-	46440	6944

Table 3: Number of function evaluation when applying spectral deferred correction with adaptive step-size to third order IVP (14) with initial step-size $k = 0.2$ and different values of m, J, and required precision.

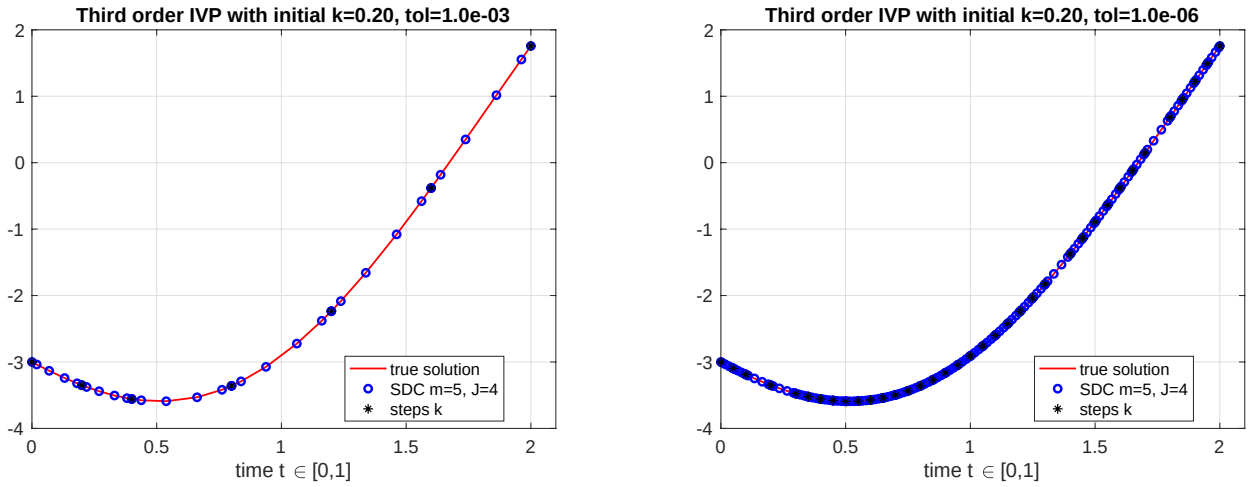


Figure 5: Solution of Plot for third order IVP (14) obtained by spectral deferred correction with $m = 5$, $J = 4$ with adaptive step-size scheme for different precisions. Circles show the computed solution on Chebyshev nodes in each subinterval. Black asterisks show the end-points of subintervals with step-size k (note that if precision is satisfied for two subintervals in a row, k is doubled). True solution is shown with red solid lines.

Similar results for Jacobi elliptic functions (15)with adaptive step-size are shown in figure 6 and table 4.

Precision	$m = 3,\ J = 2$	$m = 5,\ J = 4$	$m = 7,\ J = 6$
1e-3	192	150	280
1e-6	5808	360	280
1e-12	-	12480	1680

Table 4: Number of function evaluation when applying spectral deferred correction with adaptive step-size to Jacobi elliptic functions (15) with initial step-size $k = 0.1$ and different values of m, J, and required precision.

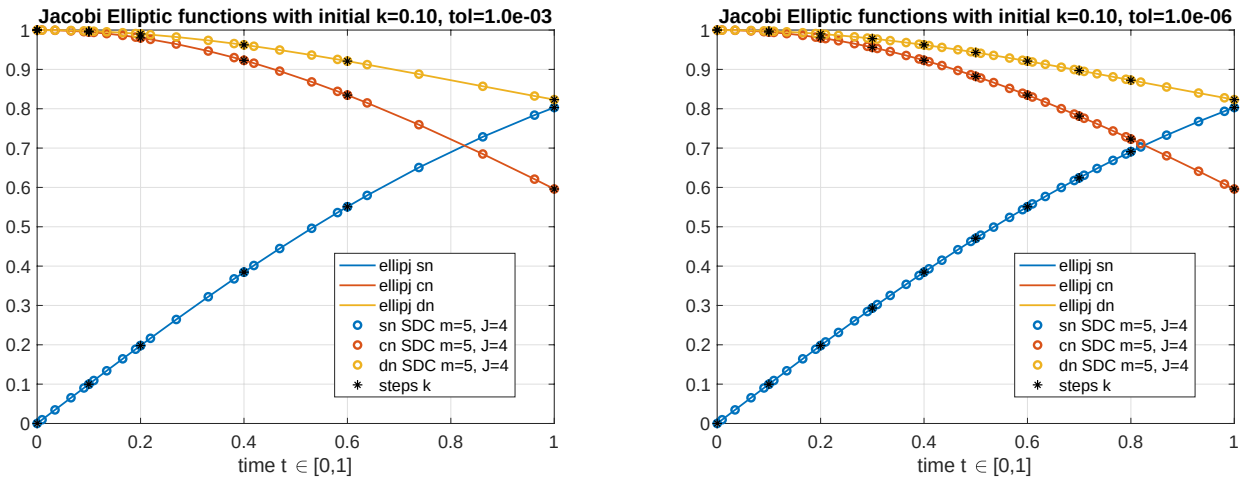


Figure 6: Solution of Jacobi elliptic functions (15) obtained by spectral deferred correction with $m = 5$, $J = 4$ with adaptive step-size scheme for different precisions. Circles show the computed solution on Chebyshev nodes in each subinterval. Black asterisks show the end-points of subintervals with step-size k (note that if precision is satisfied for two subintervals in a row, k is doubled). Reference solution $sn(t)$, $cn(t)$, $dn(t)$ from MATLAB ellipj are shown with solid lines.

5 Conclusion

In this project we reviewed the spectral deferred correction scheme for non-stiff ordinary differential equations. We implemented the spectral deferred correction method with Chebyshev nodes on subintervals. Our implementation included both fixed and adaptive step-size cases. We tested the case when step-size is fixed on two different non-stiff problem examples and showed the solution and error plots. We further investigated the effect of number of correction steps on order of accuracy for both problems. Furthermore, we showed experimental results with the adaptive step-size case in terms of number of function evaluations given a required precision. In conclusion, the spectral deferred correction method shows excellent accuracy property, is easy to implement, and only requires a low-order solver to drive the process.

6 Appendix

6.1 List of files and codes

The attached functions to this report are organized as follows.

- `Code/Demo_ChebInt.m`: This script tests the function `ChebInt, ChebFFT, iChebFFT` with a numerical example.
- `Code/Demo_FE.m`: This script implements the forward Euler method and tests it on a numerical example.
- `Code/Demo_SDC_IVP.m`: This script implements the spectral deferred correction method with fixed step-size and tests it on third order IVP (14).
- `Code/Demo_SDC_ellipj.m`: This script implements the spectral deferred correction method with fixed step-size and tests it on Jacobi elliptic functions (15).
- `Code/Demo_SDC_IVP_J.m`: This script uses the spectral deferred correction method with fix step-size and plots the effect of J for third order IVP (14).
- `Code/Demo_SDC_ellipj_J.m`: This script uses the spectral deferred correction method with fix step-size and plots the effect of J for Jacobi elliptic functions (15).
- `Code/Demo_SDCA_IVP.m`: This script implements the spectral deferred correction method with adaptive step-size scheme and solves third order IVP (14)for different precision values.
- `Code/Demo_SDCA_ellipj.m`: This script implements the spectral deferred correction method with adaptive step-size scheme and solves Jacobi elliptic functions (15)for different precision values.
- `Code/error_loglog.m`: Plots the error vs. step-size.
- `Code/error_table.m`: Prints the error vs. step-size.

6.2 Spectral integration functions

```
function [int_res] = ChebInt(f_at_r,N)
  a = ChebFFT(f_at_r); % coefficients
  % coefficients of the integral
  Tat1 = [0 , 1./([1:N])];
  Tat1(2:2:end) = -1*Tat1(2:2:end); Ta = Tat1(2:end-2) - Tat1(4:end);
  b = zeros(size(a));
  b(:,1) = a(:,1) - 0.25*a(:,2) + 0.5* sum(a(:,3:length(Ta)+2).*Ta,2);
  b(:,2) = a(:,1) - 0.5*a(:,3);
  b(:,3:end-1) = (1./[2:size(b,2)-2]).*...
                 0.5.*(a(:,2:size(b,2)-2)-a(:,4:size(b,2)));
  int_res = iChebFFT(b); % integral results
end
```

```
function [a] = ChebFFT(f)
  N = size(f,2);
  f = [f, f(:,N-1:-1:2)];
  a = real(fft(f,[],2))/(N-1);
  a = [a(:,1)/2, a(:,2:N-1), a(:,N)/2];
end

function [f] = iChebFFT(a)
  N = size(a,2);
  a = (N-1)*[a(:,1)*2, a(:,2:N-1), a(:,N)*2];
  f = ifft([a, a(:,N-1:-1:2)],[],2);
  f = f(:,1:N);
end
```

6.3 Spectral deferred correction function

```
function [u_sol, t] = FE_SDC_solver(funcf, u0, t_intval, opts)
  % Check options and assign if not given
  % opts.k: time step-size
  % opts.adaptive: use adaptive steps
  % opts.tol: tolerance for adaptive case
  % opts.m: number of cheb nodes
  % opts.J: number of SDC runs
  if ~exist('opts', 'var');
    opts.k = 0.1; opts.m = 5; opts.J = 4;
    opt.adaptive = false; opt.tol = NaN;
  else
    if ~isfield(opts, 'k'); opts.k=0.1; end
    if ~isfield(opts, 'm'); opts.m=5;   end
    if ~isfield(opts, 'J'); opts.J=4;   end
    if ~isfield(opts, 'adaptive'); opts.adaptive=false; end
    if ~isfield(opts, 'tol'); opts.tol=NaN; end
    if ~isfield(opts, 'verbose'); opts.verbose=false; end
  end
  t0 = t_intval(1); t_final = t_intval(2);
  r = cos(pi*(0:opts.m)/opts.m);
  u_sol = zeros(length(u0), 1);
  u_sol(:,1) = u0; t = t0;
  k_ind = 1; t_all = []; u_sol_all = [];
  if opts.adaptive; num_accepted_intervals = 0; end
  while t(end)<t_final
    ta = t(k_ind); tb = t(k_ind)+opts.k;
    s = fliplr(0.5*(tb-ta)*r + 0.5*(tb+ta));
    Phi = zeros(size(u_sol,1),length(s));
    Phi(:,1) = u_sol(:,k_ind);
    Phi_a = ones(size(Phi)).*Phi(:,1);
    F_phi = zeros(size(Phi));
    % run Forward Euler
    for s_ind = 1:length(s)-1
      hi = s(s_ind+1) - s(s_ind);
      F_phi(:,s_ind) = funcf(Phi(:,s_ind),s(s_ind));
      Phi(:,s_ind+1) = Phi(:,s_ind) + hi * F_phi(:,s_ind);
    end
    F_phi(:,end) = funcf(Phi(:,end),s(end));
```

```
    if opts.adaptive
        if any(Phi(:)>1e35)
          fprintf('unresolved FE observed decreasing step');
          opts.k = opts.k/2;
          if opts.k<=1e-10; error('The adaptive method failed'); end
          num_accepted_intervals = 0; continue;
        end
    end
    % loop over SDC
    for j = 1:opts.J
      % compute approximate residual function sigma
      F_int = 0.5*(tb-ta)*fliplr(ChebInt(fliplr(F_phi),opts.m));
      Sigma = F_int - Phi + Phi_a;
      % compute delta from C_exp
      F_phi_dlt = zeros(size(F_phi));
      delta = zeros(size(Phi));
      delta(:,1) = Sigma(:,1);
      for s_ind = 1:length(s)-1
              hi = s(s_ind+1)-s(s_ind);
              F_phi_dlt(:,s_ind) = funcf(Phi(:,s_ind)+...
               delta(:,s_ind),s(s_ind));
              delta(:,s_ind+1) = delta(:,s_ind) + ...
                 hi* (F_phi_dlt(:,s_ind)-F_phi(:,s_ind)) +...
                 (Sigma(:,s_ind+1)-Sigma(:,s_ind));
         end
      F_phi_dlt(:,end) = funcf(Phi(:,end)+delta(:,end),s(end));
      Phi = Phi + delta;
      F_phi = F_phi_dlt;
    end
    if opts.adaptive
      cond1 = (max(abs(delta(:)))<=opts.tol);
      ChebPhi = ChebFFT(Phi);
      ChebPhi = ChebPhi(:,end-1:end);
      cond2 = (max(abs(ChebPhi(:)))<=opts.tol);
      if (cond1 & cond2)
        num_accepted_intervals = num_accepted_intervals+1;
        t(end+1) = tb;
        u_sol(:,k_ind+1) = Phi(:,end);
        k_ind= k_ind + 1;
```

```
        if num_accepted_intervals==2
          num_accepted_intervals = 0; opts.k = opts.k*2;
          if opts.verbose; fprintf('increasing k'); end
        end
          t_all = [t_all,s]; u_sol_all = [u_sol_all,Phi]; continue;
      else
        opts.k = opts.k/2;
        if opts.verbose; fprintf('Conditions not satisfied'); end
        if opts.k<=1e-10; error('The adaptive method failed.'); end
        num_accepted_intervals = 0;
        continue;
      end
    end
    t(end+1) = tb; u_sol(:,k_ind+1) = Phi(:,end);
    t_all = [t_all,s]; u_sol_all = [u_sol_all,Phi]; k_ind= k_ind + 1;
  end
end
```

References

[DGR00] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.