We describe a new class of algorithms for the solution of parabolic partial differential equations (PDEs). This class of schemes is based on three principal observations. First, the spatial discretization of parabolic PDEs results in stiff systems of ordinary differential equations (ODEs) in time, and therefore, requires an implicit method for its solution. Spectral Deferred Correction (SDC) methods use repeated iterations of a low-order method (e.g. implicit Euler method) to generate a high-order scheme. As a result, SDC methods of arbitrary order can be constructed with the desired stability properties necessary for the solution of stiff differential equations. Furthermore, for large-scale systems, SDC methods are more computationally efficient than implicit Runge-Kutta schemes. Second, implicit methods for the solution of a system of linear ODEs yield linear systems that must be solved on each iteration. It is well known that the linear systems constructed from the spatial discretization of parabolic PDEs are sparse. In $\mathbb{R}^1$, these linear systems can be solved in $\mathcal{O}(n)$ operations where n is the number of spatial discretization nodes. However, in $\mathbb{R}^2$, the straightforward spatial discretization leads to matrices with dimensionality $n^2 \times n^2$ and bandwidth n. While fast inversion schemes of $\mathcal{O}(n^3)$ exist, we use alternating direction implicit (ADI) methods to replace the single two-dimensional implicit step with two sub-steps where only one direction is treated implicitly. This approach results in schemes with computational cost $\mathcal{O}(n^2)$. Likewise, ADI methods in $\mathbb{R}^3$ have computational cost $\mathcal{O}(n^3)$. While popular ADI methods are low-order, we combine the SDC methods with an ADI method to generate computationally efficient, high-order schemes for the solution of parabolic PDEs in $\mathbb{R}^2$ and $\mathbb{R}^3$. Third, traditional pseudospectral schemes for the representation of the spatial operator in parabolic PDEs yield differentiation operators with eigenvalues that can be excessively large. We improve on the traditional approach by subdividing the entire spatial domain, constructing bases on each subdomain, and combining the obtained discretization with the implicit SDC schemes. The resulting schemes are high-order in both time and space and have computational cost $\mathcal{O}(N \cdot M)$ where N is the number of spatial discretization nodes and M is the number of temporal nodes. We illustrate the behavior of these schemes with several numerical examples.

Spectral Deferred Corrections for Parabolic Partial Differential Equations

Daniel Beylkin[†]

Technical Report YALEU/DCS/TR-1512
June 8, 2015

[†] This author was supported in part by the DOD, AFOSR NDSEG Fellowship Award FA9550-11-C-0028, ONR Grants N00014-11-1-0718 and N00014-14-1-0797, and Telcordia Grant PO#20013660

Program in Applied Mathematics, Yale University, New Haven, CT 06511

Approved for public release: distribution is unlimited.

Keywords: *spectral deferred corrections, parabolic partial differential equations, alternating direction implicit, pseudospectral.*

1 Introduction

The diffusion equation with variable coefficients is an extremely well-studied subject. It is usually written as

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\alpha(x) \nabla \phi) , \tag{1.1}$$

where α is the diffusion coefficient and $x \in \mathbb{R}^d$ for $d = 1, 2, 3$. This equation arises in various physical phenomena, including heat transfer and fluid dynamics, biological and chemical reaction processes, financial mathematics, and many other application areas. Furthermore, the diffusion equation is an important component of many non-linear partial differential equations (PDEs), including the Navier–Stokes equations. Thus, there is continued interest in constructing accurate and fast methods for its solution. Traditional approaches for computing its numerical solution include finite difference, finite element, integral equation, spectral, and pseudospectral methods (see, e.g., [5, 7, 12, 13, 14, 17, 32, 35, 36, 40]).

Finite difference and finite element methods tend to lead to low-order schemes for the solution of (1.1) and, hence, tend to require an excessive number of nodes in both the temporal and spatial discretization. Furthermore, high-order finite difference and finite element methods have difficulty maintaining the high-order near the boundary. As a consequence, there is continued interest in developing high-order methods in both time and space.

For the case of constant coefficients, one can convert (1.1) to an integral equation and evaluate the heat potential (see, e.g., [20, 21, 22]). For variable coefficients, some of the issues of spatial discretization mentioned above are resolved in discontinuous Galerkin or discontinuous spectral element methods (see, e.g., [24, 34, 37]).

Spatial discretization of (1.1) results in a stiff system of ordinary differential equations (ODEs) in time and, therefore, requires an implicit method for its solution. While there are no implicit A-stable multistep methods of order greater than two, there is a family of implicit multistep methods which are $A(\alpha)$-stable called backward differentiation formulae (BDF). However, since there are no BDFs of order greater than six, implicit Runge-Kutta schemes are frequently used for the solution of stiff ODEs. On the other hand, implicit Runge-Kutta schemes are computationally expensive for large systems. An alternative approach, which we use in this dissertation, is the so-called Spectral Deferred Correction (SDC) methods [10]. To avoid high cost, SDC methods use repeated iterations of a low-order method (e.g., implicit Euler method) to generate a high-order scheme. As a result, SDC methods of arbitrary order can be constructed that retain the desired stability properties necessary for the solution of stiff differential equations. This makes it an appropriate tool for the solution of parabolic PDEs in time. For example, recent work using SDC for computational fluid dynamics include [28, 30, 31].

In general, implicit methods for the solution of a system of linear ODEs yield a linear system that must be solved on each iteration. It is well known that the linear system constructed from the spatial discretization of the variable coefficient operator in (1.1) is sparse. In $\mathbb{R}^1$, this linear system is either banded or block banded and can be solved in $\mathcal{O}(n)$ where n is the number of spatial discretization nodes. However, in $\mathbb{R}^2$, the straightforward spatial discretization leads to matrices with dimensionality $n^2 \times n^2$ and bandwidth n. While there exist fast inversion schemes that cost $\mathcal{O}(n^3)$, alternating direction implicit (ADI) methods replace the single two-dimensional implicit step with two sub-steps where only one direction is treated implicitly. This approach results in schemes with computational cost $\mathcal{O}(n^2)$. Likewise, ADI methods in $\mathbb{R}^3$ have computational cost $\mathcal{O}(n^3)$. However, popular ADI methods such as those by Peaceman-Rachford or Yanenko are low-order (see, e.g., [39, 40, 44]). In this dissertation, we combine the SDC methods for the solution of a system of ODEs with an ADI method to generate computationally efficient, high-order schemes for the solution of parabolic PDEs in $\mathbb{R}^2$ and $\mathbb{R}^3$.

We also construct high-order representations of the variable coefficient operator in (1.1). Traditionally, pseudospectral schemes have been used for this purpose; however, some of the eigenvalues of the resulting differentiation operator can be excessively large. An improvement on the traditional approach is to subdivide the entire spatial domain, construct bases on each subdomain, and combine the obtained discretization with the implicit SDC schemes. The resulting schemes are high-order and have CPU time requirements that are linear in the number of spatial discretization nodes. In addition to solving (1.1), this approach is a foundation for solving nonlinear equations with diffusion components as well as linear and nonlinear problems of wave propagation.

This dissertation is organized as follows. Section 2 summarizes various standard mathematical facts to be used in subsequent sections. Section 3 contains a description of the SDC methods for a system of ODEs. In Section 4, we describe low-order solution methods for parabolic PDEs and their extensions to higher dimensions. In Section 5, we discuss the high-order discretization of the spatial variable in order to construct accurate representations of the second-order differentiation matrix. In Section 6, we describe the SDC methods for parabolic PDEs and, in Section 7, illustrate the behavior of these methods with several numerical examples.

2 Preliminaries

In this section, we introduce notation and summarize several well known facts which we use in the dissertation.

2.1 Accuracy and Stability of ODE Solvers

We assume that the initial value problem to be solved is

$$\begin{aligned} &\varphi'(t) = F(t, \varphi(t)), \;\; t \in [a, b] \\ &\varphi(a) = \varphi_a \end{aligned} \tag{2.1}$$

where $\varphi(t) : \mathbb{R} \to \mathbb{C}^n$ and $F : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$ is sufficiently smooth (to permit high-order methods).

Given a numerical solution to (2.1), $\tilde{\varphi}(b)$, the numerical method is said to be of order k if, for any sufficiently smooth F, there exists a constant $M > 0$, such that

$$\|\tilde{\varphi}(b) - \varphi(b)\| < M(b-a)^{k+1}. \tag{2.2}$$

The scalar linear differential equation

$$\begin{aligned} &\varphi'(t) = \lambda\varphi(t), \;\; t \geq 0 \\ &\varphi(0) = 1, \end{aligned} \tag{2.3}$$

where $\lambda \in \mathbb{C}$, has exact solution

$$\varphi(t) = e^{\lambda t}, \tag{2.4}$$

so that for $\mathcal{R}e(\lambda) < 0$

$$\lim_{t\to\infty} \varphi(t) = 0. \tag{2.5}$$

Conceptually, a numerical method for the solution of (2.3) with $\mathcal{R}e(\lambda) < 0$ should likewise yield a solution that converges to zero. In this sense, the numerical method should exhibit the same asymptotic behavior as the exact solution (see, e.g. [26, 39]).

Any numerical method for the solution of (2.3) produces a sequence of approximations $\tilde{\varphi}_n$ to $\varphi(nh)$ for $n = 0, 1, \ldots,$ which satisfy the recurrence relation

$$\tilde{\varphi}_{n+1} = g(h\lambda)\tilde{\varphi}_n. \tag{2.6}$$

Here, h either denotes the step size for multistep schemes or the interval size for, e.g., Runge-Kutta schemes. Introducing the notation $z = h\lambda$, the stability region of the numerical method is the set of all $z \in \mathbb{C}$ such that

$$\lim_{n\to\infty} \tilde{\varphi}_n = 0. \tag{2.7}$$

In other words, the stability region (together with its boundary) is defined to be

$$\{z \in \mathbb{C} : |g(z)| \leq 1\} \tag{2.8}$$

where $g(z)$ is known as the amplification factor. If a method is stable for all z in the left-half plane (i.e., $\mathcal{R}e(z) \leq 0$), then it is said to be A-stable. A method is said to be $A(\alpha)$-stable if it is stable for all z such that

$$\pi - \alpha \leq \arg(z) \leq \pi + \alpha. \tag{2.9}$$

Thus, A-stability is $A(\alpha)$-stability with $\alpha = \frac{\pi}{2}$. Finally, if a method is A-stable, it is said to be L-stable if

$$\lim_{\mathcal{R}e(z)\to-\infty} g(z) = 0. \tag{2.10}$$

In cases when $g(z)$ is not known analytically, the stability properties of the method can be computed numerically. Specifically, the amplification factor $g(z)$ can be evaluated as

$$g(z) = \tilde{\varphi}(h). \tag{2.11}$$

See Section 3.5 for further details.

For a given ε, the accuracy region is defined as the set of all $z \in \mathbb{C}$ such that

$$\left| e^{\lambda} - \tilde{\varphi}(h) \right| < \varepsilon. \tag{2.12}$$

Consider now the system of linear differential equations

$$\begin{aligned} \psi'(t) &= A\psi(t), \;\; t \geq 0 \\ \psi(0) &= \bar{1}, \end{aligned} \tag{2.13}$$

with

$$\psi(t) = \{\psi_1(t), \ldots, \psi_d(t)\}^T, \tag{2.14}$$

and

$$\bar{1} = \{1, \ldots, 1\}^T. \tag{2.15}$$

We assume that A is a diagonalizable $d \times d$-matrix with eigenvalues $\lambda_1, \ldots, \lambda_d$ such that $\mathcal{R}e(\lambda_j) \leq 0$ for all $j = 1, \ldots, d$. Then, the exact solution of (2.13) is

$$\varphi(t) = e^{At}. \tag{2.16}$$

Any numerical method for the solution of (2.13) produces a sequence of approximations $\tilde{\psi}_n$ to $\psi(nh)$ for $n = 0, 1, \ldots$, which satisfy the recurrence relation

$$\tilde{\psi}_{n+1} = g(hA)\tilde{\psi}_n. \tag{2.17}$$

As in the scalar case, $g(z)$ is the amplification factor and the definitions for A-stability, $A(\alpha)$-stability, and L-stability are identical. In particular, the numerical method is stable if the spectral radius,

$$\rho(g(hA)) \leq 1. \tag{2.18}$$

2.2 Lagrange Interpolation

Given nodes $t_0, \ldots, t_{m-1}$, the Lagrange basis polynomials

$$\ell_j(t) = \prod_{\substack{i=0 \\ i \neq j}}^{m-1} \left(\frac{t - t_i}{t_j - t_i} \right), \quad j = 0, \ldots, m-1 \tag{2.19}$$

satisfy the property

$$\ell_j(t_i) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases} \tag{2.20}$$

Hence,

$$\psi(t) = \sum_{j=0}^{m-1} f(t_j)\ell_j(t) \tag{2.21}$$

is the Lagrange interpolating polynomial for $f(t)$, i.e., $\psi(t_j) = f(t_j)$. If f is a polynomial of degree $m-1$, then $\psi(t) = f(t)$.

Observation 1. *For evaluation of (2.21), we compute $\ell_j(t)$ as*

$$\ell_j(t) = \frac{w_j}{t - t_j}\phi(t) \tag{2.22}$$

where

$$w_j = \frac{1}{\prod_{\substack{i=0 \\ i \neq j}}^{m-1} (t_j - t_i)} \tag{2.23}$$

and

$$\phi(t) = \prod_{\substack{i=0\\i\neq j}}^{m-1} (t - t_i). \tag{2.24}$$

Thus,

$$\psi(t) = \phi(t) \sum_{j=0}^{m-1} \frac{w_j}{t - t_j} f(t_j), \tag{2.25}$$

which is known as the modified form of Lagrange's interpolation formula. The computation of each w_j costs $\mathcal{O}(m)$, so $\{w_j\}$ can be pre-computed and the total cost is $\mathcal{O}(m^2)$ operations. Once $\{w_j\}$ have been computed, the cost of evaluating $\psi(t)$ at any point t is $\mathcal{O}(m)$. Since m is $\mathcal{O}(1)$, the cost of evaluating the function $f(t)$ is $\mathcal{O}(1)$.

2.3 Legendre Polynomials

In what follows, we will denote by $P_j(x)$ the j^{th} Legendre polynomial on the interval $[-1, 1]$, defined by the three-term recurrence

$$P_{j+1}(x) = \frac{2j+1}{j+1} x P_j(x) - \frac{j}{j+1} P_{j-1}(x) \tag{2.26}$$

with the initial conditions $P_0(x) = 1$ and $P_1(x) = x$ (see, e.g., [1, 18]). Each Legendre polynomial satisfies the differential equation

$$(1 - x^2)\frac{d^2 P_j(x)}{dx^2} - 2x\frac{dP_j(x)}{dx} + j \cdot (j+1) P_j(x) = 0. \tag{2.27}$$

The family of polynomials $\{P_j\}$ is orthogonal on $[-1, 1]$, that is,

$$\int_{-1}^{1} P_i(x) P_j(x) dx = \frac{2}{2j+1} \delta_{ij}. \tag{2.28}$$

We denote by $\tilde{P}_j$ the normalized Legendre polynomials, that is,

$$\tilde{P}_j(x) = \sqrt{\frac{2j+1}{2}} P_j(x). \tag{2.29}$$

The following Lemma is well known. Its proof can be found in, e.g., [3].

Lemma 2. *$P_j(x)$ satisfies the following two relations:*

$$(2j+1) P_j(x) = P'_{j+1}(x) - P'_{j-1}(x) \tag{2.30}$$

and

$$P'_j(x) = (2j-1) P_{j-1}(x) + (2j-5) P_{j-3}(x) + (2j-9) P_{j-5}(x) + \ldots \tag{2.31}$$

for $j \geq 1$.

2.4 Gaussian Integration

The nodes and weights of Gaussian quadratures are chosen so that polynomials of degree less than or equal to $2n-1$, $\{\phi_0, \ldots, \phi_{2n-1}\}$, are integrated exactly, that is,

$$\sum_{i=0}^{n-1} w_i \phi_j(t_i) = \int_a^b \phi_j(t)dt \tag{2.32}$$

for all $j = 0, \ldots, 2n-1$. Provided that the function f is well approximated by linear combinations of such polynomials, the Gaussian quadrature formulae provide good approximations to integrals of the form

$$\int_a^b f(t)\omega(t)dx, \tag{2.33}$$

where $\omega(x)$ is an integrable non-negative function [8, 39].

The following are common choices of quadrature nodes $t_0, \ldots, t_{n-1}$ for the integration of functions defined on $[a, b] = [-1, 1]$. In all cases, $\omega(t) = 1$, except for Chebyshev nodes for which $\omega(t) = (1-t^2)^{-\frac{1}{2}}$. See, e.g., [1, 6, 8, 39] for further details.

- Gauss-Legendre: t_i is the i^{th} zero of $P_n(t)$, the n^{th}-degree Legendre polynomial.
- Chebyshev: $t_i = \cos(\frac{2i-1}{2n}\pi)$, which is the i^{th} zero of $T_n(t)$, the n^{th}-degree Chebyshev polynomial.
- Gauss-Lobatto: $t_0 = -1$, $t_{n-1} = 1$, and t_i is the i^{th} zero of $P'_{n-1}(t)$ for $i = 2, \ldots, m-2$.
- Left-hand Gauss-Radau: $t_0 = -1$ and t_i is the i^{th} zero of

$$\frac{P_{n-1}(t) + P_n(t)}{1+t}. \tag{2.34}$$

  The right-hand Gauss-Radau quadratures are the negative of the left-hand Gauss-Radau quadratures.

In this dissertation, we will also use what we call a right-hand variant of Gauss-Legendre nodes,

$$t_i = 2\frac{y_i + 1}{y_{n-1} + 1} - 1 \tag{2.35}$$

where y_i is the i^{th} zero of $P_n(t)$. We refer to it as a right-hand variant of Gauss-Legendre because $x_{n-1} = 1$.

Observation 3. *In some cases, analytical expression for the weights w_i exist. However, in general, once the nodes $t_0, \ldots, t_{n-1}$ are computed, the weights $w_0, \ldots, w_{n-1}$ can be determined by solving the linear system of equations*

$$\sum_{i=0}^{n-1} w_i \phi_j(t_i) = \int_{-1}^1 \phi_j(t)\omega(t)dt$$

for $i = 0, \ldots, n-1$.

3 Spectral Deferred Corrections

3.1 The Picard Integral Equation

Integrating the initial value problem in (2.1) with respect to t, yields the Picard integral equation

$$\varphi(t) = \varphi_a + \int_a^t F(\tau, \varphi(\tau))d\tau. \tag{3.1}$$

Suppose we have an approximate solution $\tilde{\varphi}(t)$ to (2.1). A measure of the quality of the approximation is given by the residual function

$$\varepsilon(t) = \varphi_a + \int_a^t F(\tau, \tilde{\varphi}(\tau))d\tau - \tilde{\varphi}(t). \tag{3.2}$$

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

$$\delta(t) = \varphi(t) - \tilde{\varphi}(t). \tag{3.3}$$

Substituting (3.3) back into (3.1), we obtain

$$\tilde{\varphi}(t) + \delta(t) = \varphi_a + \int_a^t F(\tau, \tilde{\varphi}(\tau) + \delta(\tau))d\tau. \tag{3.4}$$

Substituting (3.2) into (3.4) yields

$$\delta(t) = \int_a^t \left[F(\tau, \tilde{\varphi}(\tau) + \delta(\tau)) - F(\tau, \tilde{\varphi}(\tau))\right] d\tau + \varepsilon(t). \tag{3.5}$$

By defining the function $G : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$ to be

$$G(t, \delta) = F(t, \tilde{\varphi}(t) + \delta(t)) - F(t, \tilde{\varphi}(t)), \tag{3.6}$$

we can rewrite (3.5) as a Picard-type integral equation

$$\delta(t) - \int_a^t G(\tau, \delta(\tau))d\tau = \varepsilon(t). \tag{3.7}$$

This is the key equation for Spectral Deferred Correction (SDC) methods as in, e.g. [10, 27].

3.2 Implicit Euler for the Picard Equation

Suppose that we have nodes $a \leq t_0 < t_1 < \cdots < t_{m-1} \leq b$. Then, the implicit Euler scheme for the solution of (2.1), or equivalently (3.1), is given by

$$\varphi_{i+1} = \varphi_i + h_i \cdot F(t_{i+1}, \varphi_{i+1}), \;\; h_i = t_{i+1} - t_i \tag{3.8}$$

for $i = 0, \ldots, m-2$.

Similarly, the implicit Euler scheme for the solution of (3.7) is given by

$$\delta_{i+1} = \delta_i + h_i \cdot G(t_{i+1}, \varphi_{i+1}) + (\varepsilon(t_{i+1}) - \varepsilon(t_i)). \tag{3.9}$$

Observation 4. *Given an approximate solution $\tilde{\varphi}(t)$ to (2.1), a key component of the scheme in (3.9) is the evaluation of the integral in (3.2). This requires stable, high-order methods for interpolation and integration.*

3.3 Spectral Integration

We would like to evaluate the integral

$$\int_a^{t_i} f(\tau)d\tau \tag{3.10}$$

for $i = 0, \ldots, m-1$. A standard approach is to represent the function f by a truncated series expansion and compute the integral by analytically integrating the series. The result is a linear mapping between the vector of m function values $\{f(t_i)\}_{i=0}^{m-1}$ and the vector $\left\{\int_a^{t_i} f(\tau)d\tau\right\}_{i=0}^{m-1}$. This approach is numerically stable provided that the quadrature nodes are selected carefully, e.g., Gaussian-Legendre or Chebyshev nodes (see Section 2.4). In this case, the matrix representation of the linear mapping is referred to as the spectral integration matrix. Its singular values are between $\mathcal{O}(1), \ldots, \mathcal{O}(1/m^2)$. See, e.g., ([17, 19, 39]) for a more detailed discussion on spectral integration matrices and their numerical properties.

The classical approach to the construction of the spectral integration matrix is to use Lagrange interpolating polynomials defined in (2.21). Specifically, given nodes $t_0, \ldots, t_{m-1}$ and a function $f(t)$, the integral of the Lagrange interpolating polynomial,

$$\psi(t) = \sum_{j=0}^{m-1} f(t_j)\ell_j(t), \tag{3.11}$$

is

$$\int_a^{t_i} \psi(\tau)d\tau = \sum_{j=0}^{m-1} f(t_j) \int_a^{t_i} \ell_j(\tau)d\tau. \tag{3.12}$$

If we define

$$S_{ij} = \int_a^{t_i} \ell_j(\tau)d\tau, \tag{3.13}$$

then

$$\int_a^{t_i} \psi(\tau)d\tau = Sf \tag{3.14}$$

where f is a vector with elements $f_i = f(t_i)$. The matrix S is referred to as the integration matrix. If the nodes $t_0, \ldots, t_{m-1}$ are Gauss-Legendre or Chebyshev nodes, then S is referred to as the spectral integration matrix.

Observation 5. *An alternative formulation useful for the computation of the spectral integration matrix in (3.13) expresses the function $f(t)$ through Legendre polynomials instead of Lagrange interpolating polynomials. Specifically, given $\{\alpha_j\}_{j=0}^{m-1}$ so that*

$$f(t) = \sum_{j=0}^{m-1} \alpha_j P_j(t), \tag{3.15}$$

we would like to compute $\{\beta_j\}_{j=0}^{m}$ so that

$$\int_{-1}^{t} f(\tau)d\tau = \sum_{j=0}^{m} \beta_j P_j(t). \tag{3.16}$$

Denoting

$$\begin{aligned}\overline{f} &= \{f(t_0), \ldots, f(t_{m-1})\}^T \\ \overline{\alpha} &= \{\alpha_0, \ldots, \alpha_{m-1}\}^T,\end{aligned} \tag{3.17}$$

we have

$$\overline{f} = V\overline{\alpha} \tag{3.18}$$

where

$$V_{ij} = P_j(t_i). \tag{3.19}$$

To compute the indefinite integral, we have

$$\int_{-1}^{t} f(\tau)d\tau = \sum_{j=0}^{m-1} \alpha_j \int_{-1}^{t} P_j(\tau)d\tau. \tag{3.20}$$

Using (2.30), we obtain

$$\sum_{j=0}^{m-1} \alpha_j \int_{-1}^{t} P_j(\tau)d\tau = \alpha_0 \left(P_0(t) + P_1(t)\right) + \sum_{j=1}^{m-1} \frac{\alpha_j}{2j+1} \left(P_{j+1}(t) - P_{j-1}(t)\right). \tag{3.21}$$

Hence, we arrive at

$$\beta_j = \begin{cases} \alpha_0 - \frac{1}{3}\alpha_1 & j = 0 \\ \frac{\alpha_{j-1}}{2j-1} - \frac{\alpha_{j+1}}{2j+3} & 0 < j < m-1 \\ \frac{\alpha_{m-2}}{2m-3} & j = m-1 \\ \frac{\alpha_{m-1}}{2m-1} & j = m. \end{cases} \tag{3.22}$$

We denote by A, the $(m+1) \times m$-matrix such that

$$A\overline{\alpha} = \overline{\beta}, \tag{3.23}$$

where

$$\overline{\beta} = \{\beta_0, \ldots, \beta_m\}^T. \tag{3.24}$$

Finally, we denote by $\tilde{V}$ the $m \times (m+1)$-matrix such that

$$\tilde{V}_{ij} = P_j(t_i). \tag{3.25}$$

Then, it is straightforward to see that the spectral integration matrix, S, is given by

$$S = \tilde{V}AV^{-1}. \tag{3.26}$$

Given any approximate solution $\tilde{\varphi}(t)$, the residual function is

$$\varepsilon(t) = \varphi_a + \int_a^t F(\tau, \tilde{\varphi}(\tau))d\tau - \tilde{\varphi}(t). \tag{3.27}$$

Denoting

$$\begin{aligned}\overline{\varphi} &= \{\tilde{\varphi}(t_0), \ldots, \tilde{\varphi}(t_{m-1})\}^T \\ \overline{\varphi_a} &= \{\varphi_a, \ldots, \varphi_a\}^T \\ \overline{F}(\tilde{\varphi}) &= \{F(t, \tilde{\varphi}(t_0)), \ldots, F(t_{m-1}, \tilde{\varphi}(t_{m-1}))\}^T,\end{aligned} \tag{3.28}$$

we approximate the residual function by the vector

$$\sigma(\tilde{\varphi}) = \overline{\varphi_a} + S\overline{F}(\tilde{\varphi}) - \overline{\varphi} \tag{3.29}$$

where S is the spectral integration matrix.

3.4 Spectral Deferred Corrections Algorithm

Spectral Deferred Correction methods for the solution of (2.1) proceed as follows. First, an approximate solution $\varphi^0(t)$ with error $\delta(t) = \varphi(t) - \varphi^0(t)$ is computed at quadrature nodes $a \leq t_0 < t_1 < \cdots < t_{m-1} \leq b$ via some numerical method - generally, low order. Then, an approximation for the error $\delta(t)$ is computed at the quadrature nodes and the updated solution becomes $\varphi^1(t) = \varphi^0(t) + \delta(t)$. This procedure is repeated L times to provide a more accurate approximation $\varphi^L(t)$.

More explicitly, given some choice of quadrature nodes, Algorithm 1 is the implicit SDC algorithm from [10].

Algorithm 1 Implicit Spectral Deferred Corrections

```
Compute an initial approximate solution, $\varphi^0(t_j)$, for $j = 0, \dots, m-1$ via the implicit Euler method (3.8).

For $l = 1, \dots, L$
    1. Compute the residual function $\varepsilon(t_j)$ in (3.2) via spectral integration (see Section 3.3),
    2. Compute $\delta(t_j)$ via the implicit Euler scheme (3.9),
    3. Update the approximate solution $\varphi^l(t_j) = \varphi^{l-1}(t_j) + \delta(t_j)$.

Set the resulting solution as $\varphi(t_j) = \varphi^L(t_j)$.
```

Given the value of the solution $\varphi(t)$ at $t_0, \dots, t_{m-1}$, we can evaluate the solution everywhere on the interval $[a, b]$ via Lagrange interpolation (see Section 2.2).

Observation 6. *Many choices of $t_0, \dots, t_{m-1}$ do not include the right endpoint. If (2.1) is not stiff, then $\varphi(b)$ can be computed via Gaussian integration (see Section 2.4) as*

$$\varphi(b) = \varphi_a + \int_a^b F(\tau, \varphi(\tau))d\tau = \varphi_a + \sum_{i=0}^{m-1} w_i F(t_i, \varphi(t_i)). \tag{3.30}$$

For stiff systems, this approach is numerically unstable. Instead, the solution can be computed at $t = b$ via Lagrange interpolation. Nevertheless, it is for this reason that quadrature schemes that include the right end point (i.e., $t_{m-1} = b$) are preferred (e.g. right-hand Gauss-Radau or right-hand Gauss-Legendre).

See, e.g., [4, 10] for the proof of the following theorem.

Theorem 7. *For any sufficiently smooth function F, the approximate solution computed via SDC converges to the exact solution with order* $\min(m, L+1)$.

Observation 8. *Replacing the implicit Euler method with the higher-order trapezoidal method reduces the required number of corrections, but results in a scheme that is not L-stable. Moreover, such an approach requires careful implementation. For example, using*

Gauss-Legendre nodes and extrapolating the solution to $t = b$ via Lagrange interpolation can result in a scheme whose stability properties are like that of an explicit method. We avoid such concerns by using the implicit Euler method to drive the SDC algorithm.

3.5 Stability and Accuracy Regions of Spectral Deferred Corrections

In [10], Spectral Deferred Corrections using Gauss-Legendre quadratures nodes are introduced. However, the choice of the nodes strongly influences the stability and accuracy regions of the resulting schemes. In this section, we compare the stability and accuracy properties of SDC methods with four classes of nodes: Gauss-Legendre, Chebyshev, Gauss-Lobatto, and what we call a right-hand variant of Gauss-Legendre (see Section 2.4).

In Section 2.1, we introduce the concepts of stability and accuracy regions of numerical methods for the solution of ODEs. For clarity, we restate several facts here. To compare the stability and accuracy properties of the SDC methods, we apply them to the scalar linear differential equation

$$\begin{aligned} &\varphi'(t) = \lambda\varphi(t), \;\; t \geq 0 \\ &\varphi(0) = 1, \end{aligned} \tag{3.31}$$

where $\lambda \in \mathbb{C}$, has exact solution

$$\varphi(t) = e^{\lambda t}. \tag{3.32}$$

Traditionally, for a fixed time step h, the stability of the numerical method is formulated in terms of the so-called amplification factor $g(h\lambda)$ in (2.6). The stability region (together with its boundary) is defined to be

$$\{z \in \mathbb{C} : |g(z)| \leq 1\}. \tag{3.33}$$

where $z = h\lambda$. If a method is stable for all z such that $\mathcal{R}e(z) \leq 0$, then it is said to be A-stable. If the method is A-stable, it is said to be L-stable if

$$\lim_{\mathcal{R}e(z) \to -\infty} g(z) = 0. \tag{3.34}$$

Finally, for a given ε, the accuracy region is defined as the set of all $z \in \mathbb{C}$ such that

$$\left| e^{\lambda} - \tilde{\varphi}(h) \right| < \varepsilon, \tag{3.35}$$

where $\tilde{\varphi}(h)$ is the approximate solution at $t = h$ computed via some numerical method. Since the stability function of SDC methods is not known analytically, the stability region of the methods are computed numerically via Algorithm 2. A simple modification of the algorithm is used to compute the accuracy region.

If we fix $h = 1$, then a simple calculation shows that for a given λ, $\tilde{\varphi}(1) = g(\lambda) = g(z)$. Therefore, to compute the boundary of the stability region, we find λ such that $\tilde{\varphi}(1) = 1$. To start, we use the fact that for $z = 0$, $g(z) = 1$ for SDC methods. The algorithm then proceeds to take small steps and, after each step, searches in the direction perpendicular to that step to find λ such that $\tilde{\varphi}(1) = 1$.

Algorithm 2 Stability Region Boundary Search

```
Initialization:

  1. Starting from $\lambda_0 = 0$, take a small step along the positive imaginary axis and denote
     this point $\tilde{\lambda}_1$,

  2. For $\lambda = \tilde{\lambda}_1$, determine if $\tilde{\varphi}(1) > 1$ or $\varphi(1) < 1$,

  3. If $\tilde{\varphi}(1) > 1$, search parallel to the real axis in the negative direction to find $\lambda$ such
     that $\tilde{\varphi}(1) = 1$. If $\tilde{\varphi}(1) < 1$, search parallel to the real axis in the positive direction to
     find $\lambda$ such that $\tilde{\varphi}(1) = 1$. Denote this by $\lambda_1$.

For $j = 1, 2, \ldots$

  1. Starting from $\lambda_j$, take a small step in the same direction as the vector connecting
     $\lambda_{j-1}$ to $\lambda_j$ and denote this point $\tilde{\lambda}_j$. Denote the vector by $v_j = \{x_j, y_j\}^T$,

  2. For $\lambda = \tilde{\lambda}_j$, determine if $\tilde{\varphi}(1) > 1$ or $\varphi(1) < 1$,

  3. If $\tilde{\varphi}(1) > 1$, search in the direction $\{-y_j, x_j\}^T$ to find $\lambda$ such that $\tilde{\varphi}(1) = 1$. If
     $\tilde{\varphi}(1) < 1$, search in the direction $\{y_j, -x_j\}^T$ to find $\lambda$ such that $\tilde{\varphi}(1) = 1$. Denote this
     by $\lambda_j$.

Continue the iteration until $\lambda$ returns to the origin.
```

In this section, we compare SDC methods of order 4, 6, 8, 10, and 12. In each case, the number of nodes m equals to the order of the SDC method and the number of deferred corrections L equals to $m-1$. Similar analysis for other choices of nodes can be found in [29]. For a given order, we compare the stability region, the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and the accuracy region for different discretization nodes. Unless otherwise stated, in the plots of the stability region in Figures 3.1, 3.4, 3.7, 3.10, and 3.13, the lines indicate where $g(z) = 1$ with $g(z) < 1$ *outside* of the lines. In the plots of the amplification factor $g(z)$ for real z in Figures 3.2, 3.5, 3.8, 3.11, and 3.14, $x = -\log_{10}(-z)$. Thus, the x-axis in these plots, which has range $x \in [-8, 5]$, corresponds to $z \in [-10^8, -10^{-5}]$. Finally, in the plots of the accuracy region in Figures 3.3, 3.6, 3.9, 3.12, and 3.15, the lines indicate where $\varepsilon = 10^{-5}$.

In Figures 3.1, 3.2, and 3.3 we compare the stability and accuracy properties of SDC methods of order 4 for different sets of discretization nodes. Figure 3.1 is a plot of the stability regions, Figure 3.2 is a plot of the amplification factors $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and Figure 3.3 is a plot of the accuracy regions. In this case, the fourth order SDC method with Gauss-Legendre nodes is A-stable, while the rest are $A(\alpha)$-stable with α slightly less than $\frac{\pi}{2}$. For all choices of discretization nodes, $g(z) < 1$ as $\mathcal{R}e(z) \to -\infty$. However, clearly the SDC method with Gauss-Lobatto nodes has a much larger amplification factor in the limit than the other choices of nodes.

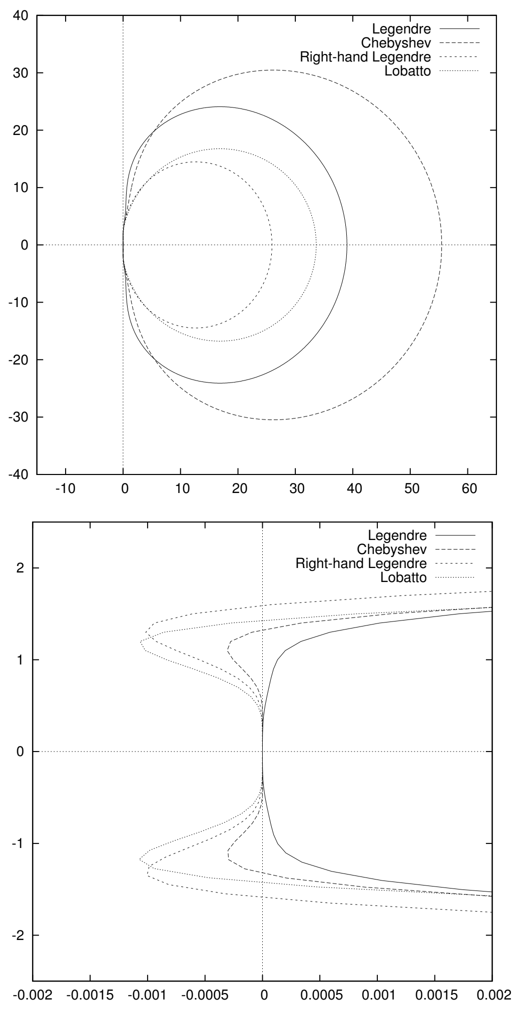

Figure 3.1: Comparison of the stability regions for fourth order SDC methods with different sets of discretization nodes. The lines indicate where the amplification factor $g(z) = 1$ with $g(z) < 1$ outside of the lines. The bottom figure is a plot of the stability regions zoomed in on the origin.

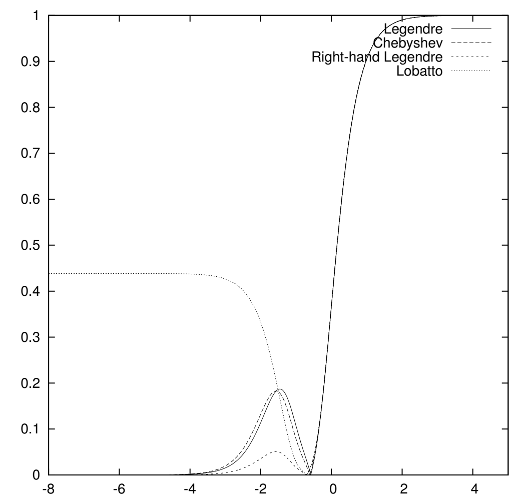

Figure 3.2: Comparison of the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$ for fourth order SDC methods with different sets of discretization nodes. In the x-axis of the plot, $x = -\log_{10}(-z)$.

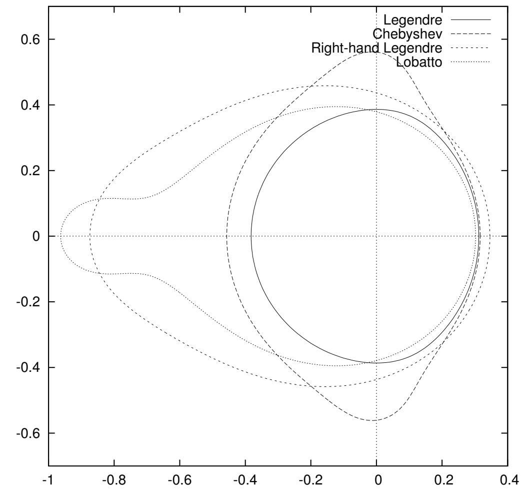

Figure 3.3: Comparison of the accuracy regions with $\varepsilon = 10^{-5}$ for fourth order SDC methods with different sets of discretization nodes.

In Figures 3.4, 3.5, and 3.6 we compare the stability and accuracy properties of SDC methods of order 6 for different sets of discretization nodes. Figure 3.4 is a plot of the stability regions, Figure 3.5 is a plot of the amplification factors $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and Figure 3.6 is a plot of the accuracy regions. In this case, none of the sixth order SDC methods are A-stable, but they all are $A(\alpha)$-stable with α slightly less than $\frac{\pi}{2}$. For all choices of discretization nodes, $g(z) < 1$ as $\mathcal{R}e(z) \to -\infty$. However, as in the fourth order case, clearly the SDC method with Gauss-Lobatto nodes has a much larger amplification factor in the limit than the other choices of nodes.

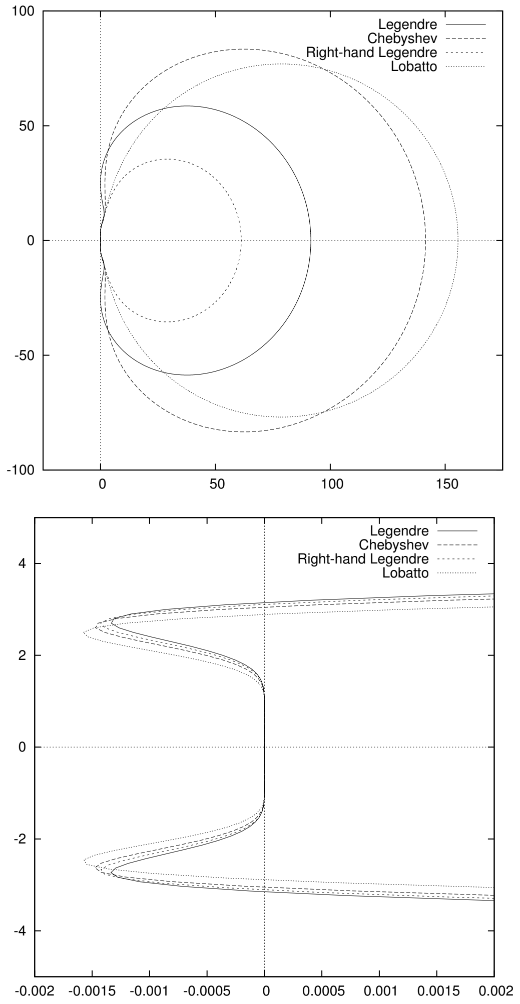

Figure 3.4: Comparison of the stability regions for sixth order SDC methods with different sets of discretization nodes. The lines indicate where the amplification factor $g(z) = 1$ with $g(z) < 1$ outside of the lines. The bottom figure is a plot of the stability regions zoomed in on the origin.

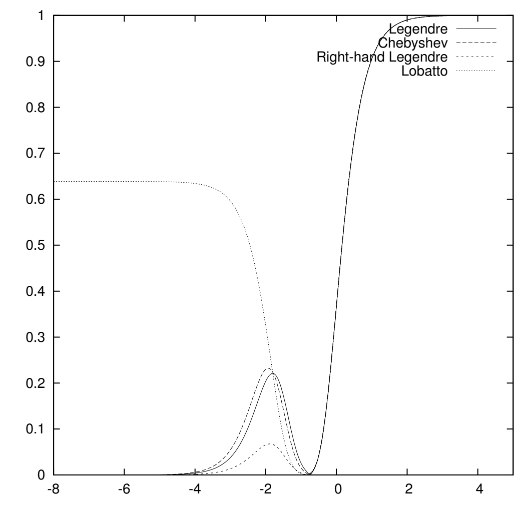

Figure 3.5: Comparison of the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$ for sixth order SDC methods with different sets of discretization nodes. In the x-axis of the plot, $x = -\log_{10}(-z)$.

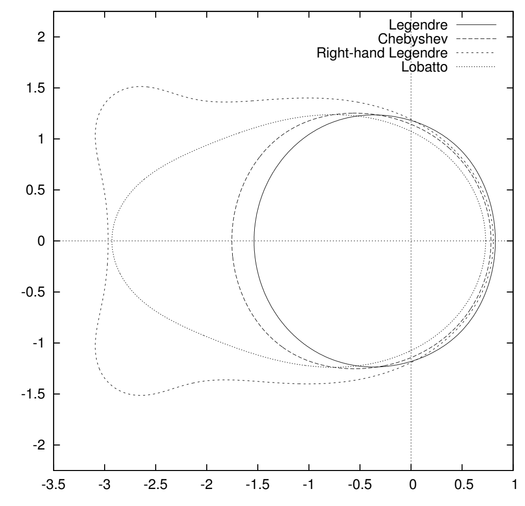

Figure 3.6: Comparison of the accuracy regions with $\varepsilon = 10^{-5}$ for sixth order SDC methods with different sets of discretization nodes.

In Figures 3.7, 3.8, and 3.9 we compare the stability and accuracy properties of SDC methods of order 8 for different sets of discretization nodes. Figure 3.7 is a plot of the stability regions, Figure 3.8 is a plot of the amplification factors $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and Figure 3.9 is a plot of the accuracy regions. In this case, none of the eighth order SDC methods are A-stable, but they all are $A(\alpha)$-stable with α slightly less than $\frac{\pi}{2}$. For all choices of discretization nodes, $g(z) < 1$ as $\mathcal{R}e(z) \to -\infty$. However, as in the fourth and sixth order cases, clearly the SDC method with Gauss-Lobatto nodes has a much larger amplification factor in the limit than the other choices of nodes.

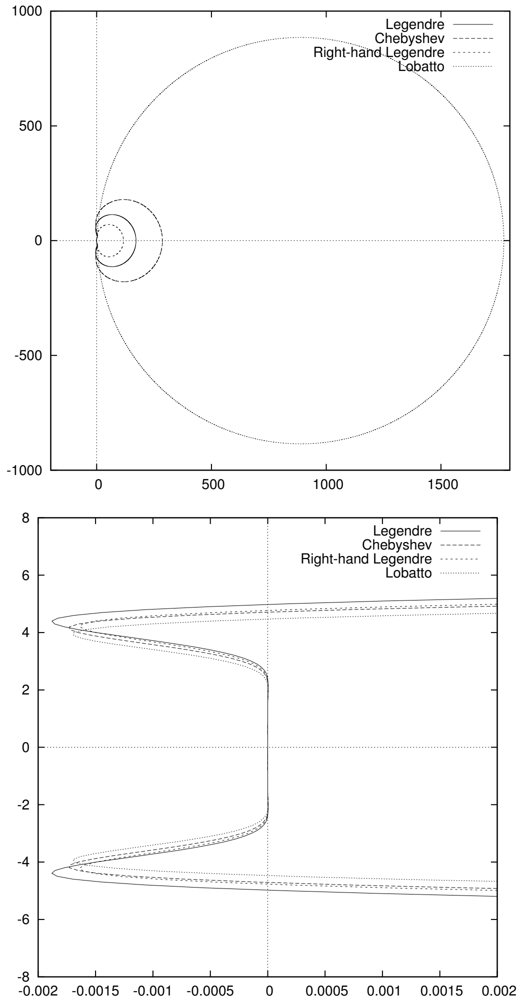

Figure 3.7: Comparison of the stability regions for eighth order SDC methods with different sets of discretization nodes. The lines indicate where the amplification factor $g(z) = 1$ with $g(z) < 1$ outside of the lines. The bottom figure is a plot of the stability regions zoomed in on the origin.

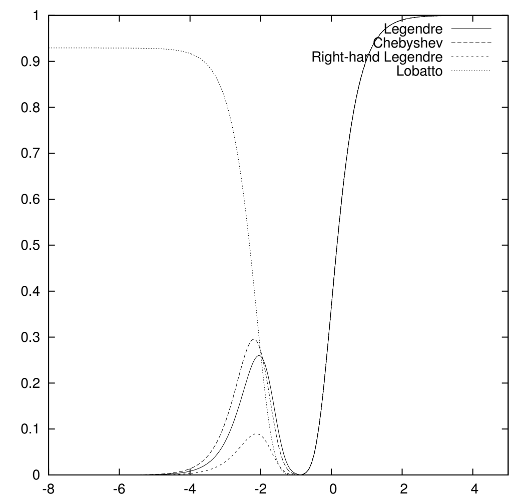

Figure 3.8: Comparison of the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$ for eighth order SDC methods with different sets of discretization nodes. In the x-axis of the plot, $x = -\log_{10}(-z)$.

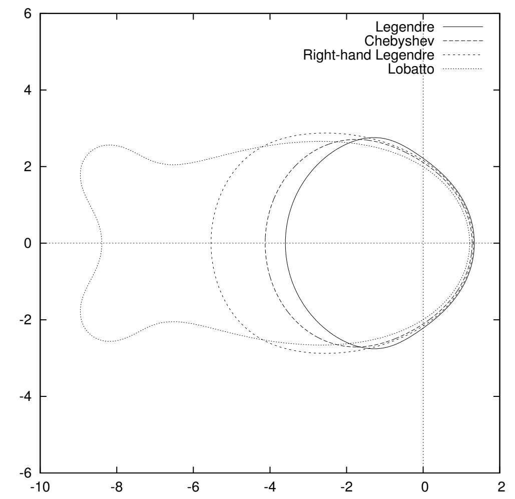

Figure 3.9: Comparison of the accuracy regions with $\varepsilon = 10^{-5}$ for eighth order SDC methods with different sets of discretization nodes.

In Figures 3.10, 3.11, and 3.12 we compare the stability and accuracy properties of SDC methods of order 10 for different sets of discretization nodes. Figure 3.10 is a plot of the stability regions, Figure 3.11 is a plot of the amplification factors $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and Figure 3.12 is a plot of the accuracy regions. In the plot of the stability regions in Figure

3.10, the lines indicate where $g(z) = 1$ with $g(z) < 1$ *outside* of the lines except for the SDC method with Gauss-Lobatto nodes where the line indicated where $g(z) = 1$ with $g(z) < 1$ *inside* of the line. In this case, none of the tenth order SDC methods are A-stable, but, with the exception of the SDC method with Gauss-Lobatto nodes, they all are $A(\alpha)$-stable with α slightly less than $\frac{\pi}{2}$. Despite the implicit construction for the SDC scheme with Gauss-Lobatto nodes, the stability region is like that of an explicit method. For all choices of discretization nodes except Gauss-Lobatto, $g(z) < 1$ as $\mathcal{R}e(z) \to -\infty$.

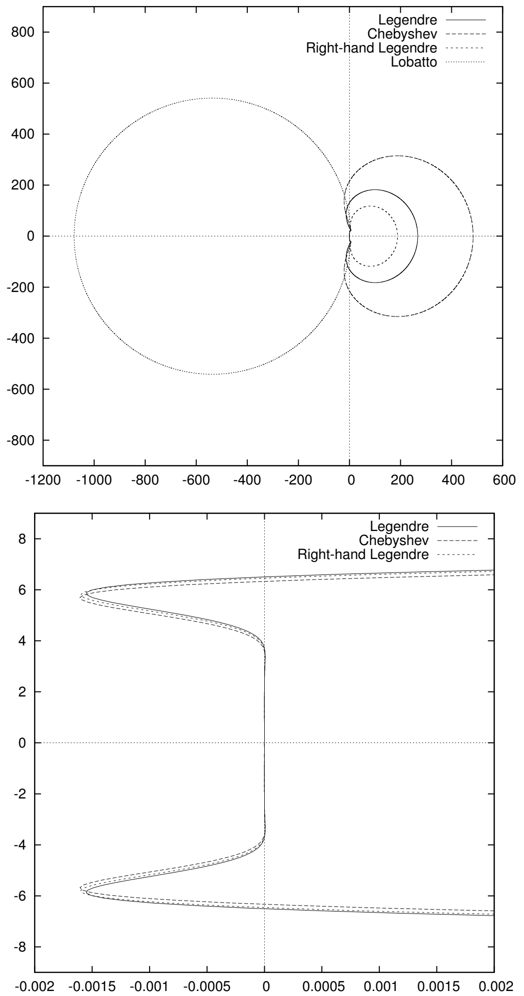

Figure 3.10: Comparison of the stability regions for tenth order SDC methods with different sets of discretization nodes. The lines indicate where the amplification factor $g(z) = 1$ with $g(z) < 1$ outside of the lines except for Gauss-Lobatto nodes for which $g(z) < 1$ inside of the line. The bottom figure is a plot of the stability regions zoomed in on the origin.

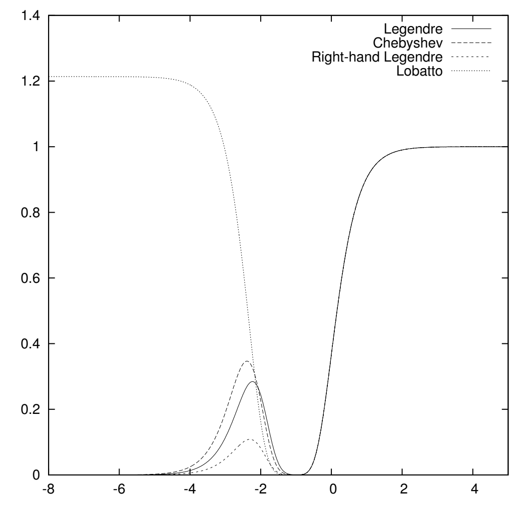

Figure 3.11: Comparison of the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$ for tenth order SDC methods with different sets of discretization nodes. In the x-axis of the plot, $x = -\log_{10}(-z)$.

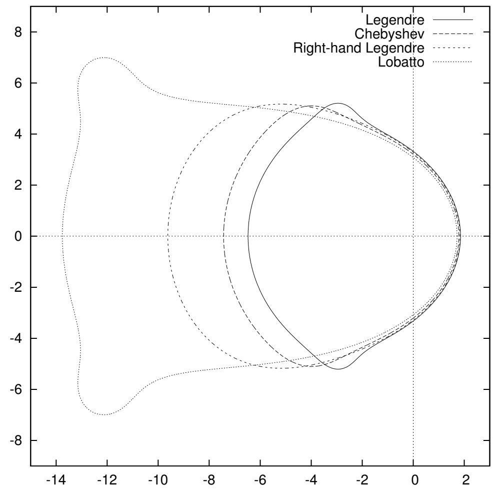

Figure 3.12: Comparison of the accuracy regions with $\varepsilon = 10^{-5}$ for tenth order SDC methods with different sets of discretization nodes.

In Figures 3.13, 3.14, and 3.15 we compare the stability and accuracy properties of SDC methods of order 12 for different sets of discretization nodes. Figure 3.13 is a plot of the stability regions, Figure 3.14 is a plot of the amplification factors $g(z)$ as $\mathcal{R}e(z) \to -\infty$, and Figure 3.15 is a plot of the accuracy regions. In the plot of the stability regions in Figure

3.13, the lines indicate where $g(z) = 1$ with $g(z) < 1$ *outside* of the lines except for the SDC method with Gauss-Lobatto nodes where the line indicated where $g(z) = 1$ with $g(z) < 1$ *inside* of the line. In this case, none of the twelfth order SDC methods are A-stable, but, with the exception of the SDC method with Gauss-Lobatto nodes, they all are $A(\alpha)$-stable with α slightly less than $\frac{\pi}{2}$. Despite the implicit construction for the SDC scheme with Gauss-Lobatto nodes, the stability region is like that of an explicit method. For all choices of discretization nodes except Gauss-Lobatto, $g(z) < 1$ as $\mathcal{R}e(z) \to -\infty$.

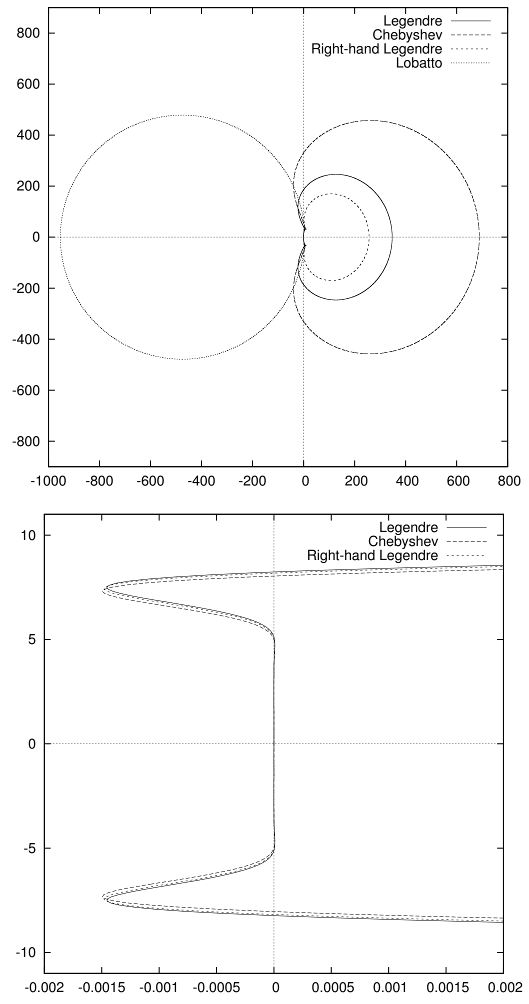

Figure 3.13: Comparison of the stability regions for twelfth order SDC methods with different sets of discretization nodes. The lines indicate where the amplification factor $g(z) = 1$ with $g(z) < 1$ outside of the lines except for Gauss-Lobatto nodes for which $g(z) < 1$ inside of the line. The bottom figure is a plot of the stability regions zoomed in on the origin.

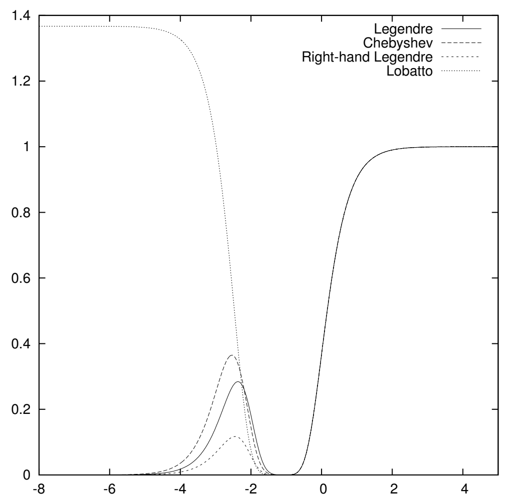

Figure 3.14: Comparison of the amplification factor $g(z)$ as $\mathcal{R}e(z) \to -\infty$ for twelfth order SDC methods with different sets of discretization nodes. In the x-axis of the plot, $x = -\log_{10}(-z)$.

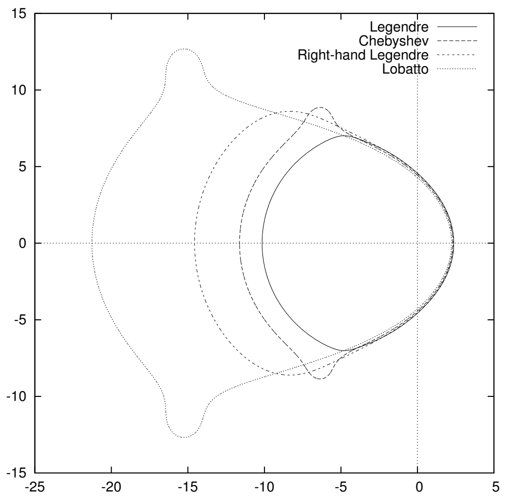

Figure 3.15: Comparison of the accuracy regions with $\varepsilon = 10^{-5}$ for twelfth order SDC methods with different sets of discretization nodes.

It is clear from the stability regions in Figures 3.10 and 3.13 that the choice of Gauss-Lobatto nodes for high-order schemes is not a practical option. Between the other choices of nodes, the differences in the stability and accuracy regions are fairly small. However, the SDC scheme with the right-hand variant of Gauss-Legendre nodes consistently had (slightly) larger accuracy regions with essentially equivalent stability properties. For this reason, in our implementation of SDC methods for evolving parabolic differential equations in time, we use the right-hand variant of Gauss-Legendre nodes.

4 Parabolic PDE Solvers

In order to develop an SDC algorithm for parabolic PDEs, we need a low-order numerical solver to drive the PDE version of the SDC method. This scheme serves the same role as that the implicit Euler method in (3.8) and (3.9) does for the implicit SDC algorithm for the solution of ODEs. In many situations, the scheme should retain the L-stability property provided by the implicit Euler method for ODEs. In this section, we discuss the implicit Euler method for parabolic PDEs in one spatial variable and its extension to higher dimensions, known as the Alternating Direction Implicit (ADI) method. Using ADI helps control the computational cost of the numerical solver. As a general reference, we point to, e.g., [32, 40, 44] for a more detailed discussion of numerical techniques for the solution of PDEs.

4.1 Implicit Euler Methods for Parabolic PDEs

Consider the parabolic PDE in one dimension

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial \phi}{\partial x}\right), \tag{4.1}$$

where α is the diffusion coefficient. Discretizing (4.1) using spatial nodes $x_0, \dots, x_{n-1}$ yields a system of ODEs

$$\frac{\partial \varphi}{\partial t} = \tilde{D}_x \left(A D_x\right) \varphi, \tag{4.2}$$

where

$$\varphi = \varphi(t) = \left\{\phi(x_0, t), \dots, \phi(x_{n-1}, t)\right\}^T. \tag{4.3}$$

Matrices D_x and $\tilde{D}_x$ are discrete approximations of the derivative operator and A is the operator of point-wise multiplication of the function $\alpha(x)$ defined via the formula

$$A(\varphi)(x_j) = \alpha(x_j)\varphi(x_j). \tag{4.4}$$

Observation 9. *Our construction of D_x and $\tilde{D}_x$ results in a matrix $\tilde{D}_x A D_x$ that is symmetric. While the matrices D_x and $\tilde{D}_x$ both approximate the derivative operator, in our construction they are not necessarily the same. In particular, the matrix D_x enforces the relevant boundary conditions, while the matrix $\tilde{D}_x$ does not. For details, see Sections 5.2 and 5.3.*

Given nodes $t_0 < \cdots < t_{m-1}$, the implicit Euler method for the solution of (4.2) is

$$\varphi^{(i+1)} = \varphi^{(i)} + h_i \tilde{D}_x A D_x \varphi^{(i+1)}, \;\; h_i = t_{i+1} - t_i \tag{4.5}$$

or

$$\varphi^{(i+1)} = (I - h_i \tilde{D}_x A D_x)^{-1} \varphi^{(i)}. \tag{4.6}$$

The following Lemma is an immediate consequence of the definition of stability in (2.18) and L-stability in (2.10).

Lemma 10. *The implicit Euler method is stable provided*

$$\rho\left((I - h_i \tilde{D}_x A D_x)^{-1}\right) \leq 1 \tag{4.7}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix. Hence, it is stable if $\tilde{D}_x A D_x$ is negative semi-definite. Finally, the implicit Euler method is also L-stable.

Observation 11. *For most practical implementations of the implicit Euler method for parabolic PDEs, the matrix $\tilde{D}_x A D_x$ is banded or block banded. As as result, it is possible to efficiently solve linear systems with the matrix $I - h_i \tilde{D}_x A D_x$. In this case, the computational cost of solving such linear systems is $\mathcal{O}(n)$. For parabolic PDEs in $\mathbb{R}^2$, the discretized spatial grid is $n \times n$ and, therefore, the matrix representation of the Laplacian is $n^2 \times n^2$ - see Figure 4.1. In this case, there exist fast inversion schemes that cost $\mathcal{O}(n^3)$ (see, e.g., [9, 15, 25]). However, we instead use an alternating direction implicit scheme (see, e.g., [39, 40]), which is much simpler to implement and more computationally efficient. Using an alternating direction implicit scheme is even more advantageous in $\mathbb{R}^3$.*

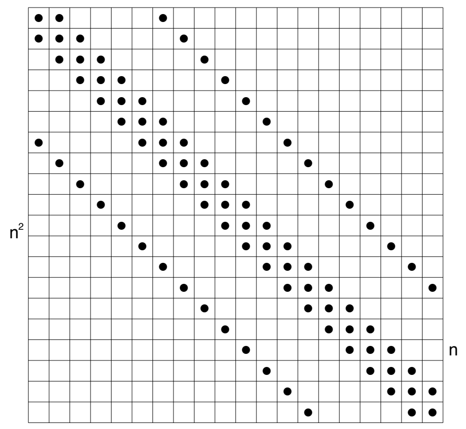

Figure 4.1: Matrix representation of the Laplacian in $\mathbb{R}^2$.

4.2 Alternating Direction Implicit Methods for Parabolic PDEs

Consider the parabolic PDE in two dimensions

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\alpha(x, y) \nabla \phi). \tag{4.8}$$

Discretizing (4.8) using spatial nodes $x_0, \ldots, x_{n-1}$ and $y_0, \ldots, y_{n-1}$ yields a system of ODEs

$$\frac{\partial \varphi}{\partial t} = \left(\tilde{D}_x (A D_x) + \tilde{D}_y (A D_y)\right) \varphi, \tag{4.9}$$

where

$$\varphi = \varphi(t) = \phi(x_{j_x}, y_{j_y}, t) \tag{4.10}$$

for all $j_x = 0, \ldots, n-1$ and $j_y = 0, \ldots, n-1$. The $n^2 \times n^2$-matrices $\tilde{D}_x A D_x$ and $\tilde{D}_y A D_y$ are discrete approximations of the second-order differentiation operator with variable coefficients. For details, see Sections 4.1, 5.2 and 5.3. In (4.9), A is the operator of point-wise multiplication of the function $\alpha(x, y)$ defined via the formula

$$A(\varphi)(x_{j_x}, y_{j_y}) = \alpha(x_{j_x}, y_{j_y})\varphi(x_{j_x}, y_{j_y}). \tag{4.11}$$

Conceptually, ADI schemes in $\mathbb{R}^2$ replace the single two-dimensional implicit step with two sub-steps where only one direction is treated implicitly. This approach replaces a single $n^2 \times n^2$ system with n systems of size $n \times n$ on each sub-step. Since each system can be solved with computational cost $\mathcal{O}(n)$ (see Observation 11), the total computational cost of ADI schemes in $\mathbb{R}^2$ are $\mathcal{O}(n^2)$, with a small constant. Likewise, in $\mathbb{R}^3$, the total computational cost is $\mathcal{O}(n^3)$.

Given nodes $t_0, \ldots, t_{m-1}$, a popular ADI method is the so-called Peaceman-Rachford ADI method (see, e.g., [39, 40, 44]),

$$\begin{aligned}
\varphi^{(i+\frac{1}{2})} &= \varphi^{(i)} + \frac{h_i}{2}\tilde{D}_x A D_x \varphi^{(i+\frac{1}{2})} + \frac{h_i}{2}\tilde{D}_y A D_y \varphi^{(i)} \\
\varphi^{(i+1)} &= \varphi^{(i+\frac{1}{2})} + \frac{h_i}{2}\tilde{D}_x A D_x \varphi^{(i+\frac{1}{2})} + \frac{h_i}{2}\tilde{D}_y A D_y \varphi^{(i+1)}.
\end{aligned} \tag{4.12}$$

We rewrite (4.12) in the form

$$\varphi^{(i+1)} = L\varphi^{(i)}, \tag{4.13}$$

where

$$\begin{aligned}
L =& (I - \frac{h_i}{2}\tilde{D}_x A D_x)^{-1}(I + \frac{h_i}{2}\tilde{D}_x A D_x) \circ \\
& (I - \frac{h_i}{2}\tilde{D}_y A D_y)^{-1}(I + \frac{h_i}{2}\tilde{D}_y A D_y).
\end{aligned} \tag{4.14}$$

The following Lemma is an immediate consequence of the definition of stability in (2.18) and L-stability in (2.10). See, e.g., [39, 40, 44] for further details.

Lemma 12. *The Peaceman-Rachford ADI method is stable provided*

$$\rho(L) \leq 1. \tag{4.15}$$

Hence, it is stable if is stable if both $\tilde{D}_x A D_x$ and $\tilde{D}_y A D_y$ are negative semi-definite.

While the $n^2 \times n^2$-matrices $I - \frac{h_i}{2}\tilde{D}_x A D_x$ and $I - \frac{h_i}{2}\tilde{D}_y A D_y$ are sparse, their inverses are not. However, the computational cost of solving linear systems with these matrices is $\mathcal{O}(n^2)$ - see Observation 11.

While the straightforward analogue of the Peaceman-Rachford ADI method in $\mathbb{R}^3$ is not guaranteed to be stable even if $\tilde{D}_x A D_x$, $\tilde{D}_y A D_y$, and $\tilde{D}_z A D_z$ are negative semi-definite, stable versions in $\mathbb{R}^3$ have been constructed by Douglas, Rachford, Gunn and others (see, e.g., [40, 44] and references therein).

Observation 13. *Using (4.14) and the definition of L-stability in (2.10), it is clear that the Peaceman-Rachford ADI method is not L-stable, that is*

$$\lim_{\mathcal{R}e(z)\to-\infty} g(z) \neq 0, \tag{4.16}$$

where $g(z)$ is the amplification factor. While, strictly speaking, L-stability is not necessary for the solution of parabolic PDEs, strong suppression of the amplification factor at infinity is, that is

$$\lim_{\mathcal{R}e(z)\to-\infty} g(z) \ll 1. \tag{4.17}$$

Therefore, the Peaceman-Rachford ADI methods is not an appropriate choice to drive the SDC schemes for the solution of parabolic PDEs.

However, there exists a class of ADI methods for the solution of (4.8) based on the implicit Euler method that are L-stable (see, e.g., [44]). While these methods exhibit convergence of order one in contrast to the Peaceman-Rachford ADI methods which exhibit convergence of order two, we accelerate the convergence by using them to drive the SDC schemes.

The implicit Euler-based ADI method for the solution of (4.8) is

$$\begin{aligned} \varphi^{*} &= \varphi^{(i)} + h_i \tilde{D}_x A D_x \varphi^{*} \\ \varphi^{(i+1)} &= \varphi^{*} + h_i \tilde{D}_y A D_y \varphi^{(i+1)}. \end{aligned} \tag{4.18}$$

Its generalization to $\mathbb{R}^3$ is

$$\begin{aligned} \varphi^{*} &= \varphi^{(i)} + h_i \tilde{D}_x A D_x \varphi^{*} \\ \varphi^{**} &= \varphi^{*} + h_i \tilde{D}_y A D_y \varphi^{**} \\ \varphi^{(i+1)} &= \varphi^{**} + h_i \tilde{D}_z A D_z \varphi^{(i+1)}. \end{aligned} \tag{4.19}$$

Since we have

$$\varphi^{(i+1)} = (I - h_i \tilde{D}_y A D_y)^{-1} (I - h_i \tilde{D}_x A D_x)^{-1} \varphi^{(i)} \tag{4.20}$$

in $\mathbb{R}^2$, the method is stable provided

$$\rho\left((I - h_i \tilde{D}_y A D_y)^{-1} (I - h_i \tilde{D}_x A D_x)^{-1} \right) \leq 1. \tag{4.21}$$

The stability analysis in $\mathbb{R}^3$ is identical.

As a direct consequence of the definition of stability in (2.18) and L-stability in (2.10), we have the following Lemma. See, e.g., [44] for further details.

Lemma 14. *The implicit Euler-based ADI method is stable if both $\tilde{D}_x A D_x$ and $\tilde{D}_y A D_y$ are negative semi-definite. The implicit Euler-based ADI method is also L-stable.*

As a result, this method is an appropriate choice to drive the SDC schemes for the solution of parabolic PDEs.

5 High-Order Differentiation Matrices

In order to develop high order methods for solving PDEs, it is necessary to develop an accurate discrete approximation to the second-derivative operator with variable coefficients, $\tilde{D}_x A D_x$. In constructing $\tilde{D}_x A D_x$, we would like to obtain a symmetric, negative definite matrix for symmetric, negative-definite problems coming from linear, second-order PDEs. Furthermore, such matrix should permit fast algorithms for solving associated linear systems.

To construct the discrete approximation to the derivative operator, we first represent a function f by a truncated series expansion (e.g. Fourier or Legendre) and compute the derivative by analytically differentiating the series. The result is a linear mapping between the vector of m function values $\{f(x_i)\}_{i=0}^{n-1}$ to the vector $\{f'(x_i)\}_{i=0}^{n-1}$. Such approach is numerically stable provided that the quadrature nodes are selected carefully. For example, if $f(x)$ is periodic, we use equally-spaced nodes. If $f(x) = 0$ on the boundary of the domain (homogenous Dirichlet boundary conditions), then we use Gauss-Legendre or, alternatively, Chebyshev nodes. In this case, the matrix representation of the linear mapping between function values and values of the derivative is referred to as the spectral differentiation matrix.

For details on the construction of differentiation matrices for periodic boundary conditions, see, e.g., [23, 33]. For details on the construction of high-order differentiation matrices for homogeneous Dirichlet boundary conditions, see, e.g., [2, 16, 38, 41, 43]. We summarize the main details here.

5.1 Periodic Boundary Conditions

In order to describe discrete approximations of the derivative operator for a periodic function, we first introduce the Discrete Fourier Transform (DFT). The DFT is a transformation $\mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$ defined by

$$\hat{f}_\ell = \frac{1}{n} \sum_{j=0}^{n-1} f_j e^{-2\pi i \ell j/n}, \ \ \ell = 0, \ldots, n-1. \tag{5.1}$$

The inverse DFT is given by the formula

$$f_j = \sum_{k=0}^{n-1} \hat{f}_\ell e^{2\pi i \ell j/n}, \ \ j = 0, \ldots, n-1 \tag{5.2}$$

(see, e.g., [23, 33]). The n complex numbers $\{\hat{f}_\ell\}$ are referred to as the Fourier coefficients of the function values $\{f_j\}$. In many applications, such as those of this dissertation, it is convenient to view $\{f_j\}$ as n equally-spaced samples of a continuous, periodic real function f with a single period defined on $[0, 1]$.

Observation 15. *The DFT and its inverse can also be described by the closely related formulae*

$$\hat{f}_\ell = \frac{1}{n} \sum_{j=0}^{n-1} f_j e^{-2\pi i \ell j/n}, \ \ \ell = -\frac{n-1}{2}, \ldots, \frac{n-1}{2} \tag{5.3}$$

$$f_j = \sum_{\ell=-\frac{n-1}{2}}^{\frac{n-1}{2}} \hat{f}_\ell e^{2\pi i \ell j/n}, \;\; j = 0, \ldots, n-1. \tag{5.4}$$

While the forms (5.1) and (5.2) are the standard representation for the DFT, the forms (5.3) and (5.4) are usually preferred in applications of DFT to analysis (see, e.g., [11]). If $x_j = \frac{j}{n}$, $j = 0, \ldots, n-1$ *are the equally-spaced samples on* $[0,1]$*, then (5.4) can be restated as*

$$f(x_j) = \sum_{\ell=-\frac{n-1}{2}}^{\frac{n-1}{2}} \hat{f}_\ell e^{2\pi i \ell t_j}, \;\; j = 0, \ldots, n-1. \tag{5.5}$$

The corresponding trigonometric polynomial for the evaluation of $f(x)$ *anywhere on* $[0,1]$ *is, therefore,*

$$f(x) = \sum_{\ell=-\frac{n-1}{2}}^{\frac{n-1}{2}} \hat{f}_\ell e^{2\pi i \ell x}. \tag{5.6}$$

Obviously, straightforward computation of the DFT or its inverse costs $\mathcal{O}(n^2)$ operations. Algorithms that perform such computations in $\mathcal{O}(n \log n)$ are known as Fast Fourier Transforms (FFT) (see, e.g., [23, 33]).

Given a function $f(x)$ defined as

$$f(x) = \sum_{\ell=-\frac{n-1}{2}}^{\frac{n-1}{2}} \hat{f}_\ell e^{2\pi i \ell x}, \tag{5.7}$$

the derivative is computed as

$$f'(x) = \sum_{\ell=-\frac{n-1}{2}}^{\frac{n-1}{2}} 2\pi i \ell \hat{f}_\ell e^{2\pi i \ell x}. \tag{5.8}$$

Therefore, given a periodic function $f(x)$, with a single period defined on $[0,1]$, we compute its derivative by first computing its Fourier coefficients $\hat{f}_\ell$ via the FFT, multiplying the Fourier coefficients by a factor $2\pi i \ell$, and then applying the inverse FFT to obtain $f'(x)$.

5.2 Dirichlet Boundary Conditions: Spectral Derivative on a Single Interval

In Section 3.3, we used Legendre polynomials for the purpose of constructing the spectral integration matrix. We will now use the normalized Legendre polynomials $\{\tilde{P}_j\}$ in (2.29) to construct the spectral differentiation matrix. Since the family of polynomials $\{\tilde{P}_j\}$ is orthonormal, given

$$f(x) = \sum_{j=0}^{k-1} \alpha_j \tilde{P}_j(x), \tag{5.9}$$

we have

$$\alpha_j = \int_{-1}^{1} f(x)\tilde{P}_j(x)dx. \tag{5.10}$$

Lemma 16. *Suppose that $f(x) : [-1,1] \to \mathbb{R}$ is given by the expansion in (5.9) and $f'(x) : [-1,1] \to \mathbb{R}$ is defined as*

$$f'(x) = \sum_{j=0}^{k-1} \beta_j \tilde{P}_j(x). \tag{5.11}$$

Then

$$\overline{\beta} = B\overline{\alpha} \tag{5.12}$$

where

$$\overline{\alpha} = \{\alpha_0, \dots, \alpha_{k-1}\}^T, \tag{5.13}$$

$$\overline{\beta} = \{\beta_0, \dots, \beta_{k-1}\}^T, \tag{5.14}$$

B is the $k \times k$-matrix such that

$$B_{ij} = \tilde{P}_j(1)\tilde{P}_i(1) - \tilde{P}_j(-1)\tilde{P}_i(-1) - K_{ij}, \tag{5.15}$$

and K is the $k \times k$-matrix such that

$$K_{ij} = \int_{-1}^{1} \tilde{P}_j(x)\tilde{P}_i'(x)dx. \tag{5.16}$$

Proof. We have

$$\beta_i = \int_{-1}^{1} f'(x)\tilde{P}_i(x)dx. \tag{5.17}$$

After integrating by parts, we obtain

$$\begin{aligned} \beta_i &= f(1)\tilde{P}_i(1) - f(-1)\tilde{P}_i(-1) - \int_{-1}^{1} f(x)\tilde{P}_i'(x)dx \\ &= f(1)\tilde{P}_i(1) - f(-1)\tilde{P}_i(-1) - \sum_{j=0}^{k-1} \alpha_j \int_{-1}^{1} \tilde{P}_j(x)\tilde{P}_i'(x)dx. \end{aligned} \tag{5.18}$$

Substituting in

$$f(1) = \sum_{j=0}^{k-1} \alpha_j \tilde{P}_j(1) \tag{5.19}$$

and

$$f(-1) = \sum_{j=0}^{k-1} \alpha_j \tilde{P}_j(-1), \tag{5.20}$$

the result follows. □

Using (2.28), (2.29), and (2.31), a simple calculation shows that

$$K_{ij} = \sqrt{2j+1}\sqrt{2i+1} \tag{5.21}$$

for $j = i-1, i-3, i-5, \dots$.

Observation 17. *Given quadrature nodes $x_0, \dots, x_{k-1}$, quadrature weights $w_0, \dots, w_{k-1}$, and denoting*

$$\overline{f} = \left\{\sqrt{w_0} f(x_0), \dots, \sqrt{w_{k-1}} f(x_{k-1})\right\}^T, \tag{5.22}$$

$$\overline{\alpha} = \{\alpha_0, \dots, \alpha_{k-1}\}^T, \tag{5.23}$$

we have

$$\overline{f} = V\overline{\alpha} \tag{5.24}$$

where

$$V_{ij} = \sqrt{w_i}\tilde{P}_j(x_i). \tag{5.25}$$

It is straightforward to see that the spectral differentiation matrix, D, is given by

$$D = VBV^{-1}. \tag{5.26}$$

That is,

$$\overline{g} = D\overline{f} \tag{5.27}$$

where

$$\overline{g} = \left\{\sqrt{w_0} f'(x_0), \dots, \sqrt{w_{k-1}} f'(x_{k-1})\right\}^T. \tag{5.28}$$

If $x_0, \dots, x_{k-1}$ are Gauss-Legendre nodes and $w_0, \dots, w_{k-1}$ the corresponding weights, then it is clear from (5.9) that

$$\sqrt{w_i} f(x_i) = \sqrt{w_i} \sum_{j=0}^{k-1} \alpha_j \tilde{P}_j(x_i). \tag{5.29}$$

Likewise, it follows from the discretization of the integral in (5.10) that

$$\alpha_j = \sum_{i=0}^{k-1} \sqrt{w_i}\tilde{P}_j(x_i)\sqrt{w_i} f(x_i). \tag{5.30}$$

If we denote by U the matrix with elements

$$U_{ji} = \sqrt{w_i}\tilde{P}_j(x_i), \tag{5.31}$$

then (5.29) and (5.30) imply that $U = V^{-1}$. A simple calculation shows that V is orthogonal, i.e., $U = V^T$.

The second-order differentiation matrix

$$D^2 = VB^2U \tag{5.32}$$

obtained from (5.26) maps the values of the function $f(x)$ at $x_0, \dots, x_{k-1}$ to its second derivative $f''(x)$ at the same quadrature nodes. In order to enforce homogeneous Dirichlet boundary conditions within numerical PDE solvers, we first construct an augmented second-order differentiation matrix by adding quadrature nodes at $x = -1$ and $x = 1$, i.e., we use Gauss-Lobatto nodes ($x_0, \dots, x_{k-1}$ are therefore the interior Gauss-Lobatto nodes). We denote by $\tilde{D}^2$ a particular instance of the second-order differentiation matrix D^2 computed at the Gauss-Lobatto nodes. Since $f(-1) = f(1) = 0$, we denote by D_0^2 the second-order differentiation matrix obtained by removing the first and last columns and rows of $\tilde{D}^2$. Such construction of D_0^2 results in a second-order differentiation matrix that is symmetric and negative definite (see, e.g., [16, 43]).

Observation 18. *It is well known that differentiation is not a numerically robust procedure. Indeed, the maximum eigenvalue of the spectral differentiation matrix D is of the order $\mathcal{O}(k^2)$. This implies that the maximum eigenvalue of D^2 and, similarly, D_0^2 is of the order $\mathcal{O}(k^4)$. This presents a significant problem in computation as it limits the number of nodes that can be used in the spatial discretization. See, e.g., [16, 41, 42, 43], for more details on the spectral properties of spectral differentiation matrices.*

5.3 Dirichlet Boundary Conditions: Composite Spectral Derivative

Since the spectral differentiation matrix in D has a maximum eigenvalue of order $\mathcal{O}(k^2)$ (see Observation 18 above), in practical computations we are limited in the number of nodes, k. An alternative approach is to fix k and instead subdivide the interval. Then, we represent the function f by a truncated series expansion on each subinterval and compute the derivative by analytically differentiating the series. It turns out that such an approach yields a discrete approximation to the second-derivative operator with variable coefficients, $\tilde{D}_x A D_x$, that is block five diagonal (or block tridiagonal in certain implementations), symmetric, and negative definite, which permits us to efficiently solve linear systems with the matrix $I - h_i \tilde{D}_x A D_x$ (see, e.g., [2, 38]).

Specifically, let $[-1,1]$ be subdivided into 2^n intervals of equal size. For clarity, we assume that the number of points per subinterval k is constant. We define $f(x) : [-1,1] \to \mathbb{R}$ to be

$$f(x) = \sum_{m=0}^{2^n-1} \sum_{j=0}^{k-1} \alpha_{jm} \tilde{P}_{jm}(x) \tag{5.33}$$

where

$$\tilde{P}_{jm} = 2^{\frac{n}{2}} \sqrt{\frac{2j+1}{2}} P_j(2^n(x - \overline{x}_m) - 1) \tag{5.34}$$

and

$$\overline{x}_m = 2^{1-n} m - 1 \tag{5.35}$$

for $j = 0, \ldots, k-1$ and $m = 0, \ldots, 2^n - 1$, which has support on the interval $[\overline{x}_m, \overline{x}_{m+1}]$. The functions $\tilde{P}_{jm}(x)$ have disjoint support for different choices of m. A simple calculation demonstrates that for a fixed m, the functions $\{\tilde{P}_{jm}(x)\}$ are orthonormal for all j. This is summarized in the following Lemma.

Lemma 19. *The family of polynomials $\{\tilde{P}_{jm}(x)\}$ is orthonormal on the interval $[-1,1]$. Hence, given $f(x)$ as defined in (5.33),*

$$\alpha_{jm} = \int_{\overline{x}_m}^{\overline{x}_{m+1}} \tilde{P}_{jm}(x) f(x) dx. \tag{5.36}$$

Observation 20. *Suppose that $x_0, \ldots x_{k-1}$ are Gauss-Legendre nodes, $w_0, \ldots, w_{k-1}$ the corresponding weights, and we denote*

$$x_{il} = \overline{x}_l + 2^{1-n} (\frac{x_i + 1}{2}) \tag{5.37}$$

for all $i = 0, \dots, k-1$ and $l = 0, \dots, 2^n - 1$. Then, due to (5.33),

$$\sqrt{w_i} f(x_{il}) = \sqrt{w_i} \sum_{m=0}^{2^n-1} \sum_{j=0}^{k-1} \alpha_{jm} \tilde{P}_{jm}(x_{il}). \tag{5.38}$$

Likewise, discretizing (5.36) yields

$$\alpha_{jm} = \frac{1}{2^n} \sum_{i=0}^{k-1} \sqrt{w_i} \tilde{P}_{jm}(x_{il}) \sqrt{w_i} f(x_{il}) \tag{5.39}$$

for all $j = 0, \dots, k-1$ and $m = 0, \dots, 2^n - 1$.

Introducing the notation

$$\overline{f}_{i+k\cdot l} = \sqrt{w_i} f(x_{il}) \tag{5.40}$$

and

$$\overline{\alpha}_{j+k\cdot m} = \alpha_{jm} \tag{5.41}$$

where k is the number of points per subinterval, we observe that

$$\overline{f} = V\overline{\alpha} \tag{5.42}$$

and

$$\overline{\alpha} = U\overline{f} \tag{5.43}$$

where V and U are the $2^n k \times 2^n k$ block-diagonal matrices with entries

$$V_{i+k\cdot l, j+k\cdot m} = \sqrt{w_i} \tilde{P}_{jm}(x_{il}) \tag{5.44}$$

and

$$U_{j+k\cdot m, i+k\cdot l} = \frac{1}{2^n} \sqrt{w_i} \tilde{P}_{jm}(x_{il}), \tag{5.45}$$

respectively. By construction, we have $U = V^{-1}$.

Given $f(x)$ defined in (5.33), we would like to compute the same type of expansion for $f'(x)$. We start with the following lemma.

Lemma 21. *Given $f(x)$ as defined in (5.33),*

$$f'(x) = \sum_{m=0}^{2^n-1} \sum_{j=0}^{k-1} \beta_{jm} \tilde{P}_{jm}(x), \tag{5.46}$$

where

$$\beta_{il} = \sqrt{2^n} \Big(f(\bar{x}_{l+1}) \tilde{P}_i(1) - f(\bar{x}_l) \tilde{P}_i(-1) \Big) - 2^n \sum_{j=0}^{k-1} K_{ij} \alpha_{jl} \tag{5.47}$$

and K_{ij} is defined in (5.16) or, equivalently, in (5.21).

Proof. By Lemma 19, we have that

$$\beta_{il} = \int_{\overline{x}_l}^{\overline{x}_{l+1}} \tilde{P}_{il}(x) f'(x) dx. \tag{5.48}$$

Integrating (5.48) by parts, we obtain

$$\beta_{il} = f(x)\tilde{P}_{il}(x)|_{\bar{x}_l}^{\bar{x}_{l+1}} - \int_{\bar{x}_l}^{\bar{x}_{l+1}} f(x)\tilde{P}'_{il}(x) dx, \tag{5.49}$$

and, substituting (5.33) for $f(x)$, arrive at

$$\begin{aligned}\beta_{il} = &\sqrt{2^n}\left(f(\bar{x}_{l+1})\tilde{P}_i(1) - f(\bar{x}_l)\tilde{P}_i(-1)\right)\\ &- \sum_{m=0}^{2^n-1}\sum_{j=0}^{k-1} \alpha_{jm} \int_{\bar{x}_l}^{\bar{x}_{l+1}} \tilde{P}_{jm}(x)\tilde{P}'_{il}(x) dx.\end{aligned} \tag{5.50}$$

Due to (5.34), $\tilde{P}_{jm}(x)$ is only nonzero on the interval $[\bar{x}_l, \bar{x}_{l+1}]$ if $m = l$. Therefore,

$$\beta_{il} = \sqrt{2^n}\left(f(\bar{x}_{l+1})\tilde{P}_i(1) - f(\bar{x}_l)\tilde{P}_i(-1)\right) - 2^n \sum_{j=0}^{k-1} K_{ij}\alpha_{jl}. \tag{5.51}$$

$\square$

Observation 22. *The value of $f(x)$ in (5.33) at interior subinterval boundary values $\overline{x}_l$ for $l = 1, \dots, 2^n - 2$ is not well-defined. In particular, an interior subinterval boundary value $\overline{x}_l$ is shared by two subintervals with different Legendre expansions. Generally, the value at $\overline{x}_l$ computed using one Legendre expansion will differ from the value computed using the other one. Specifically, for $l \neq 0$,*

$$f(\bar{x}_l) = \sqrt{2^n} \sum_{j=0}^{k-1} \alpha_{j,l-1}\tilde{P}_j(1) \tag{5.52}$$

from the left subinterval and

$$f(\bar{x}_l) = \sqrt{2^n} \sum_{j=0}^{k-1} \alpha_{jl}\tilde{P}_j(-1) \tag{5.53}$$

from the right subinterval. Similarly, for $l \neq 2^n - 1$,

$$f(\bar{x}_{l+1}) = \sqrt{2^n} \sum_{j=0}^{k-1} \alpha_{jl}\tilde{P}_j(1) \tag{5.54}$$

from the left subinterval and

$$f(\bar{x}_{l+1}) = \sqrt{2^n} \sum_{j=0}^{k-1} \alpha_{j,l+1}\tilde{P}_j(-1) \tag{5.55}$$

from the right subinterval.

To approximate the interior boundary values, we use a combination of the value from the left and from the right. In other words, for some $0 \le a, b \le 1$, we define

$$f(\bar{x}_{l+1}) = \sqrt{2^n} \sum_{j=0}^{k-1} \Big((1-a) \cdot \alpha_{jl} \tilde{P}_j(1) + a \cdot \alpha_{j,l+1} \tilde{P}_j(-1) \Big) \tag{5.56}$$

and

$$f(\bar{x}_{l}) = \sqrt{2^n} \sum_{j=0}^{k-1} \Big((1-b) \cdot \alpha_{jl} \tilde{P}_j(-1) + b \cdot \alpha_{j,l-1} \tilde{P}_j(1) \Big) . \tag{5.57}$$

The choice of a and b impacts the resulting approximation to the derivative operator, which we discuss further in Section 5.3.1.

We summarize the results in the following theorem.

Theorem 23. *Suppose that $f(x)$ is given by the expansion*

$$f(x) = \sum_{m=0}^{2^n-1} \sum_{j=0}^{k-1} \alpha_{jm} \tilde{P}_{jm}(x) \tag{5.58}$$

subject to homogeneous Dirichlet boundary conditions

$$f(-1) = f(1) = 0. \tag{5.59}$$

Then

$$f'(x) = \sum_{m=0}^{2^n-1} \sum_{j=0}^{k-1} \beta_{jm} \tilde{P}_{jm}(x), \tag{5.60}$$

where

$$\beta_{il} = 2^n \sum_{j=0}^{k-1} \Big(D_{ij}^{(1)} \alpha_{j,l-1} + D_{ij}^{(0)} \alpha_{jl} + D_{ij}^{(-1)} \alpha_{j,l+1} \Big) \tag{5.61}$$

for $l = 1, \ldots, 2^n - 2$,

$$\beta_{i,0} = 2^n \sum_{j=0}^{k-1} \Big((1-a) \tilde{P}_i(1) \tilde{P}_j(1) - K_{ij} \Big) \alpha_{j,0} + a \tilde{P}_i(1) \tilde{P}_j(-1) \alpha_{j,1}, \tag{5.62}$$

and

$$\beta_{i,2^n-1} = 2^n \sum_{j=0}^{k-1} \Big(-(1-b) \tilde{P}_i(-1) \tilde{P}_j(-1) - K_{ij} \Big) \alpha_{j,2^n-1} - b \tilde{P}_i(-1) \tilde{P}_j(1) \alpha_{j,2^n-2} \tag{5.63}$$

with

$$\begin{aligned} D_{ij}^{(1)} &= -b \tilde{P}_i(-1) \tilde{P}_j(1), \\ D_{ij}^{(0)} &= (1-a)\, \tilde{P}_i(1) \tilde{P}_j(1) - (1-b)\, \tilde{P}_i(-1) \tilde{P}_j(-1) - K_{ij}, \\ D_{ij}^{(-1)} &= a \tilde{P}_i(1) \tilde{P}_j(-1). \end{aligned} \tag{5.64}$$

for some $0 \le a, b \le 1$.

For a given a and b in (5.56) and (5.57), we denote by B the $2^n k \times 2^n k$ matrix such that

$$B\overline{\alpha} = \overline{\beta} \tag{5.65}$$

where

$$\overline{\beta}_{j+k\cdot m} = \beta_{jm}$$

and k is the number of points per subinterval. As a result, the spectral differentiation matrix is given by

$$D = VBU. \tag{5.66}$$

Observation 24. *Given a sufficiently smooth function $f(x)$, the error of the approximation of $f'(x)$ constructed in Theorem 23 is $\mathcal{O}(h^{k-1})$ where $h = 2^{-n}$ is the length of the subintervals (see [2] for details).*

5.3.1 Second-Order Differentiation Matrix

The construction of the second-order differentiation matrix depends on the choice of value for $f(x)$ at the interior subinterval boundaries. In our numerical experiments, we set the value at the interior boundary as the average of the value computed from the neighboring intervals (i.e., $a = b = \frac{1}{2}$ in Theorem 23). We denote the resulting spectral differentiation matrix D_c. Then, D_c^2 is the second-order differentiation matrix that maps the values of the function $f(x)$ at quadrature nodes x_{il} to its second derivative, $f''(x)$, at the same quadrature nodes.

Our approach for the construction of the second-order differentiation matrix extends the construction on a single interval in Section 5.2 and, e.g., [16, 41, 42, 43], to multiple intervals. This approach differs from the construction used in [2]. Specifically, in order to enforce homogeneous Dirichlet boundary conditions within a numerical PDE solver, we first construct an augmented second-order differentiation matrix by adding quadrature nodes at $x = -1$ and $x = 1$. That is, on the left most boundary, we use left-hand Gauss-Radau nodes and on the right most boundary, we use right-hand Gauss-Radau nodes. We compute the second-order differentiation matrix at the augmented nodes, which we denote by $\tilde{D}_c^2$. Since $f(-1) = f(1) = 0$, we denote by $D_{c,0}^2$ the second-order differentiation matrix obtained by removing the first and last columns and rows of $\tilde{D}_c^2$.

The extra nodes are added to the outside subintervals in order to construct a Legendre series expansion with $k+1$ terms (as compared to k terms on the interior subintervals). On the outside subintervals, we project the Legendre polynomials onto the space of polynomials that satisfy the appropriate boundary condition. If the extra term in the Legendre series expansion is not added, then the basis formed by projecting onto the space of polynomials that satisfy the appropriate boundary condition will not span that entire space. This, in turn, results in a spurious zero eigenvalue that is avoided by our construction.

Observation 25. *Our construction of $D_{c,0}^2$ results in a second-order differentiation matrix that is block five diagonal, symmetric, negative definite. The condition number of $D_{c,0}^2$ is $\mathcal{O}(n_x^2 k^4)$, where $n_x = 2^n$ is the number of intervals. While the analytic condition number of the second-order differentiation matrix is $n_x^2 k^2$, our construction results in a condition number that is a factor of $\mathcal{O}(k^2)$ greater. However, since k is $\mathcal{O}(1)$, this is a numerically useful discretization of the second-order differentiation operator.*

Observation 26. *Given a function $f(x)$ at quadrature nodes $\{x_{il}\}_{i=0,\ldots,k-1}^{l=0,\ldots,2^n-1}$, we apply the matrix $D_{c,0}^2$ to the vector $\overline{f}$ to obtain approximate values for $f''(x)$ at the quadrature nodes. If $f(x)$ is sufficiently smooth, then the approximation error is $\mathcal{O}(h^{k-2})$ where $h = 2^{-n}$ is the length of the subintervals. This follows directly from Observation 24.*

Observation 27. *Due to the block banded structure of $D_{c,0}^2$, the computational cost of solving linear systems with the matrix $I - h_i D_{c,0}^2$ is $\mathcal{O}(n_x k \cdot k^2)$, which is essentially linear in the number of spatial discretization nodes.*

5.3.2 Accuracy of Approximation

The classical choice of basis for representing functions satisfying homogeneous Dirichlet boundary conditions on $[-1, 1]$ is $\{\sin(n\frac{\pi}{2}(x + 1))\}_{n\in\mathbb{Z}}$. In Figures 5.1, 5.2, and 5.3, we illustrate the accuracy in computing the second derivative of the function

$$\sin(b\frac{\pi}{2}(x+1)) \tag{5.67}$$

for varying b on the interval $x \in [-1, 1]$. Because the function does not necessarily satisfy homogeneous Dirichlet boundary conditions, we evaluate the second derivative by applying the augmented second-order differentiation matrix $\tilde{D}_c^2$. In Figures 5.1, 5.2, and 5.3, we construct the second-order differentiation matrix using 64, 256, and 1024 points, respectively. In each case, we compare the accuracy of the second-order differentiation matrices constructed with varying number of intervals and points per interval. Tables 1, 2, and 3 display the corresponding largest singular values and the condition number of the associated second-order differentiation matrix with boundary conditions enforced. It is clear that, in each case, there is a tradeoff between the accuracy of the approximation and the condition number of the matrix. For example, in modest size problems, such as that in Figure 5.1 where 64 nodes are used, the difference between the condition number for the second-order differentiation matrix with one interval and with eight intervals is only a factor of about 10. On the other hand, the range of b for which the accuracy of the approximation is better than 10^{-11} is much larger for the second-order differentiation matrix with one interval. In larger problems, such as that in Figure 5.3, where 1024 nodes are used, the condition number for the second-order differentiation matrix with one interval is large. This results in a significant loss of numerical precision that necessitates constructing the second-order differentiation matrix with many subintervals.

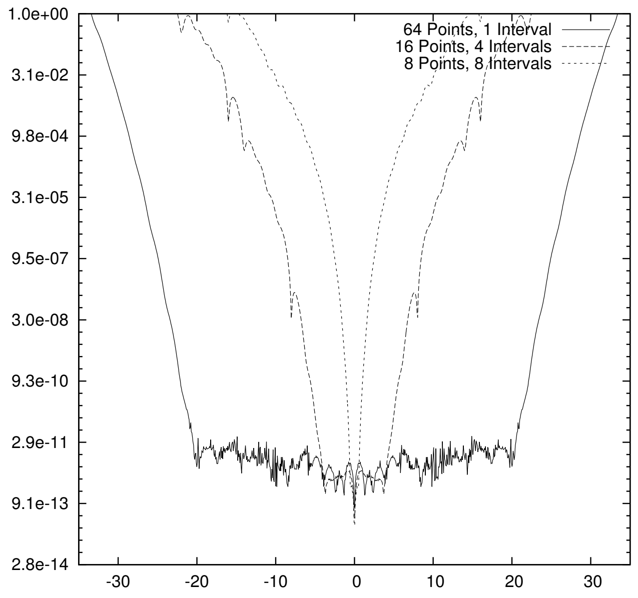

Figure 5.1: Comparison of the L^2-error for second derivative of $\sin(b\pi\frac{x+1}{2})$, where $b \in [-35, 35]$, using 64-point stencils.

Points Per Interval	Number of Intervals	Largest Singular Value of $\tilde{D}_c^2$	Condition Number of D_c^2
64	1	$1.3107 \cdot 10^6$	$1.8923 \cdot 10^5$
16	4	$1.2749 \cdot 10^5$	$5.1217 \cdot 10^4$
8	8	$3.9033 \cdot 10^4$	$1.5053 \cdot 10^4$

Table 1: Condition number of second derivative using 64-point stencils.

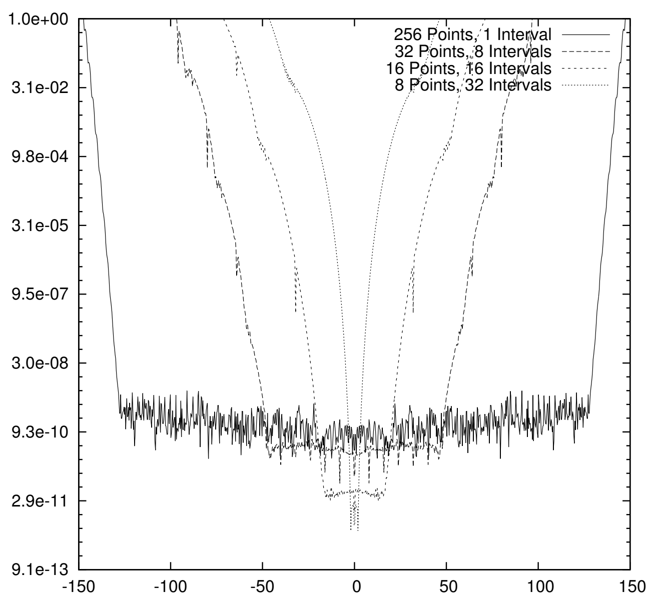

Figure 5.2: Comparison of the L^2-error for second derivative of $\sin(b\pi\frac{x+1}{2})$, where $b \in [-150, 150]$, using 256-point stencils.

Points Per Interval	Number of Intervals	Largest Singular Value of $\tilde{D}_c^2$	Condition Number of D_c^2
256	1	$3.1264 \cdot 10^8$	$4.5139 \cdot 10^7$
32	8	$7.3824 \cdot 10^6$	$2.9846 \cdot 10^6$
16	16	$2.0217 \cdot 10^6$	$8.0991 \cdot 10^5$
8	32	$6.2428 \cdot 10^5$	$2.4056 \cdot 10^5$

Table 2: Condition number of second derivative using 256-point stencils.

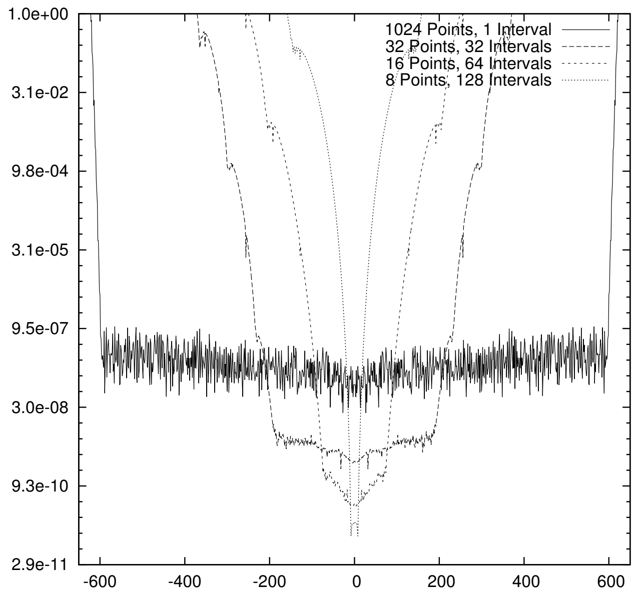

Figure 5.3: Comparison of the L^2-error for second derivative of $\sin(b\pi\frac{x+1}{2})$, where $b \in [-650, 650]$, using 1024-point stencils.

Points Per Interval	Number of Intervals	Largest Singular Value of $\tilde{D}_c^2$	Condition Number of D_c^2
1024	1	$7.8638 \cdot 10^{10}$	$1.1354 \cdot 10^{10}$
32	32	$1.1810 \cdot 10^{8}$	$4.7741 \cdot 10^{7}$
16	64	$3.2348 \cdot 10^{7}$	$1.2959 \cdot 10^{7}$
8	128	$9.9885 \cdot 10^{6}$	$3.8490 \cdot 10^{6}$

Table 3: Condition number of second derivative using 1024-point stencils.

5.3.3 Why Special Functions?

Although widely suggested, in many practical environments the choice of sine basis is inappropriate for the numerical solution of PDEs with homogeneous Dirichlet boundary conditions. In particular, the sine basis forces all even derivatives of the function to satisfy homogeneous Dirichlet boundary conditions. For example, the function

$$f(x) = \cos(\pi(x+1)) - \cos(2\pi(x+1)) \tag{5.68}$$

satisfies homogeneous Dirichlet boundary conditions but does not admit an accurate sine series representation. Specifically, while the second derivative of the sine series vanishes on the boundary, $f''(x) = 3\pi^2$ for $x = -1$ and $x = 1$. This means that the sine series representation of $f''(x)$ is not accurate near the boundary, resulting in the so-called Gibbs phenomenon. The sine series coefficients of $f''(x)$ are

$$\int_{-1}^{1} f''(x)\sin(n\frac{\pi}{2}(x+1)) = -\frac{6\pi n^3(\cos(nx)-1)}{(n^2-16)(n^2-4)} \tag{5.69}$$

for odd n. That is, the coefficients decay like $\sim \frac{1}{n}$ and are not absolutely convergent. This results in a sine series representation that converges (slowly) at every point except on the boundary.

On the other hand, in Figure 5.4 we use the example in (5.68) to investigate the numerical order of convergence for the second-order differentiation matrix (with enforced boundary conditions) $D^2_{c,0}$ with different number of quadrature nodes per subinterval. While the error initially decays exponentially fast, eventually the error for each of the schemes increases slightly due to the growing condition number of the second-order differentiation matrix.

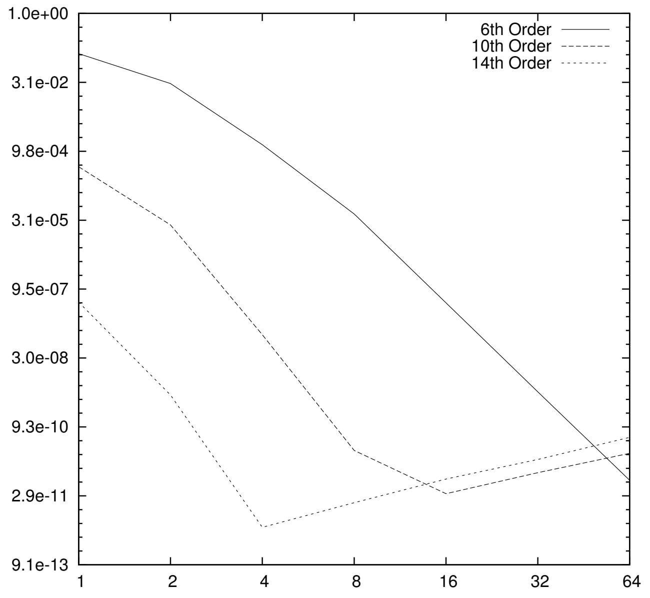

Figure 5.4: Comparison of the L^2-error in computing the second derivative of (5.68) for different order methods as a function of the number of subintervals.

6 Spectral Deferred Corrections for Parabolic PDEs

6.1 In $\mathbb{R}^1$

The variable coefficient diffusion equation in $\mathbb{R}^1$ is

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial \phi}{\partial x}\right). \tag{6.1}$$

The Spectral Deferred Corrections algorithm for the solution of (6.1) proceeds very similarly to Spectral Deferred Corrections for ODEs (see Algorithm 1). Discretizing (6.1) using spatial discretization nodes $x_0, \ldots, x_{n-1}$ yields a system of ODEs in (4.2),

$$\frac{\partial \varphi}{\partial t} = \tilde{D}_x (A D_x) \varphi, \tag{6.2}$$

which is then solved using Algorithm 1.

Algorithm 3 Spectral Deferred Corrections for Parabolic PDEs in $\mathbb{R}^1$

Compute an initial approximate solution, $\varphi^0(x_i, t_j)$, for $i = 0, \ldots, n-1$ and $j = 0, \ldots, m-1$ via the implicit Euler method (4.5) for a chosen discrete approximation to the derivative operator $\tilde{D}_x A D_x$ (see Section 5.3.1).

For $l = 1, \ldots, L$

1. Compute the residual function $\varepsilon(x_i, t_j)$ in (3.2) via spectral integration (see Section 3.3),
2. Compute $\delta(x_i, t_j)$ via the implicit Euler scheme (3.9) - see also Section 4.1,
3. Update the approximate solution $\varphi^l(x_i, t_j) = \varphi^{l-1}(x_i, t_j) + \delta(x_i, t_j)$.

Set the resulting solution as $\varphi(x_i, t_j) = \varphi^L(x_i, t_j)$.

We investigate the computational cost of Algorithm 3. We assume the number of subintervals in the spatial discretization is n_x and the number of points per subinterval is k, that is $n = n_x k$. The cost of constructing the linear system within the implicit Euler scheme is $\mathcal{O}(n_x k^2)$ and the cost of solving it is $\mathcal{O}(n_x k^3)$ (see Observation 27). In our implementation, we use a version of LU decomposition designed for block banded matrices. Since the LU decomposition of the matrix must be computed at every time step, the cost of constructing the initial approximation $\varphi^0(x_i, t_j)$ is $\mathcal{O}(n_x k^3 m)$.

We now look at the cost of a single run of spectral deferred corrections. The computation of the residual function $\varepsilon(x_i, t_j)$ involves applying the $m \times m$ spectral integration matrix to a vector representing $F(t, \varphi(t))$ at every point in space. Hence, the cost of computing the residual function is $\mathcal{O}(n_x k m^2)$. The computation of $\delta(x_i, t_j)$ involves solving the same linear systems that appeared in the construction of the initial approximation. As a result, if we store the LU decomposition of the linear system at each point in time, the cost of solving the system is $\mathcal{O}(n_x k^2)$. This has to be done at every time step, resulting in a cost of $\mathcal{O}(n_x k^2 m)$. Finally, the cost of updating the approximate solution is $\mathcal{O}(n_x k m)$.

If we run L iterations of spectral deferred corrections, then the cost is $\mathcal{O}(n_x k m^2 L) + \mathcal{O}(n_x k^2 m L)$. Therefore, the overall computational cost is

$$\mathcal{O}(n_x k \cdot m \cdot k^2) + \mathcal{O}(n_x k \cdot m \cdot mL) + \mathcal{O}(n_x k \cdot m \cdot kL). \tag{6.3}$$

Since, in practice, the values of k, m, and L are roughly equal to the desired order of the scheme, the computational cost scales linearly in the number of discretization nodes in space

$(n_x k)$, linearly in the number of temporal nodes (m), and quadratically in the desired order of the scheme $(k^2,\ mL,\ kL)$.

6.2 In $\mathbb{R}^2$ and $\mathbb{R}^3$

In $\mathbb{R}^2$ and $\mathbb{R}^3$, the algorithm is identical to Algorithm 3, except we replace implicit Euler method with the implicit Euler-based ADI method in (4.18) and (4.19).

We investigate the computational cost for the algorithm in $\mathbb{R}^2$. For simplicity, we assume the number of subintervals in the spatial discretization is n_x in both the x- and y-direction and the number of points per subinterval is k in both directions as well. Thus, the cost of constructing the linear system is $\mathcal{O}(n_x^2 k^3)$ in both the x- and y-directions. The cost of solving the resulting linear systems is $\mathcal{O}(n_x^2 k^4)$ (see Observation 27). In our implementation we used a version of LU Decomposition designed for block banded matrices. Since the LU decomposition of the matrix must be computed at every time step, the cost of constructing the initial approximation is $\mathcal{O}(n_x^2 k^4 m)$.

We now look at the cost of a single run of spectral deferred corrections. The computation of the residual function ε involves applying the $m \times m$ spectral matrix to a vector representing $F(t, \varphi(t))$ at every point in space. Hence, the cost of computing the residual function is $\mathcal{O}(n_x^2 k^2 m^2)$. The computation of δ involves solving the same linear systems that appeared in the construction of the initial approximation. As a result, if we store the LU decomposition of the linear system at each point in time, the cost of solving the system is $\mathcal{O}(n_x k^2)$ for each point in the x-direction. The cost is the same for each point in the y-direction. Hence, the total cost is $\mathcal{O}(n_x^2 k^3)$. This has to be done at every time step, resulting in a cost of $\mathcal{O}(n_x^2 k^3 m)$. Finally, the cost of updating the approximate solution is $\mathcal{O}(n_x^2 k^2 m)$.

If we run L iterations of spectral deferred corrections, then the cost is $\mathcal{O}(n_x^2 k^2 m^2 L) + \mathcal{O}(n_x^2 k^3 m L)$. Therefore, the overall computational cost is

$$\mathcal{O}(n_x^2 k^2 \cdot m \cdot k^2) + \mathcal{O}(n_x^2 k^2 \cdot m \cdot mL) + \mathcal{O}(n_x^2 k^2 \cdot m \cdot kL). \tag{6.4}$$

A similar analysis shows that the overall computational cost in $\mathbb{R}^3$ is

$$\mathcal{O}(n_x^3 k^3 \cdot m \cdot k^2) + \mathcal{O}(n_x^3 k^3 \cdot m \cdot mL) + \mathcal{O}(n_x^3 k^3 \cdot m \cdot kL). \tag{6.5}$$

As is the case in $\mathbb{R}^1$, k, m, and L are roughly equal to the desired order of the scheme. Thus, the computational cost scales linearly in the number of discretization nodes in space ($n_x^2 k^2$ in $\mathbb{R}^2$ and $n_x^3 k^3$ in $\mathbb{R}^3$), linearly in the number of temporal nodes (m), and quadratically in the desired order of the scheme $(k^2,\ mL,\ kL)$.

7 Numerical Examples

In this section we present the results of several numerical experiments in which we compute the solution to

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\alpha(x) \nabla \phi) \tag{7.1}$$

for different choices of boundary conditions and diffusion coefficients α.

7.1 Periodic Boundary Conditions with Constant Coefficients in $\mathbb{R}^1$

We compute the solution for the diffusion equation with constant coefficients

$$\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2} \;\; x \in [0, 2\pi] \tag{7.2}$$

with initial condition

$$\phi(x, 0) = \cos(10x) - 0.5\sin(8x) \tag{7.3}$$

and boundary conditions

$$\phi(0, t) = \phi(2\pi, t) \;\text{ and }\; \frac{\partial}{\partial x}\phi(0, t) = \frac{\partial}{\partial x}\phi(2\pi, t). \tag{7.4}$$

In this example, we set $\alpha = 1$.

In this case, the exact solution is

$$\phi(x, t) = \cos(10x)e^{-100t} - 0.5\sin(8x)e^{-64t}. \tag{7.5}$$

Figure 7.1 is a plot of the solution $\phi(x, t)$ in space for different points in time. In Figure 7.2 we plot the decay of the L^2-error of the solution at $t = 0.1$ as the number of subintervals in the temporal domain increases. We compare schemes of order 4, 8, and 12. In each case, the number of nodes in time m is equal to the order of the scheme and the number of deferred corrections L is equal to $m - 1$. In all cases, we use 32 equally-spaced spatial discretization nodes.

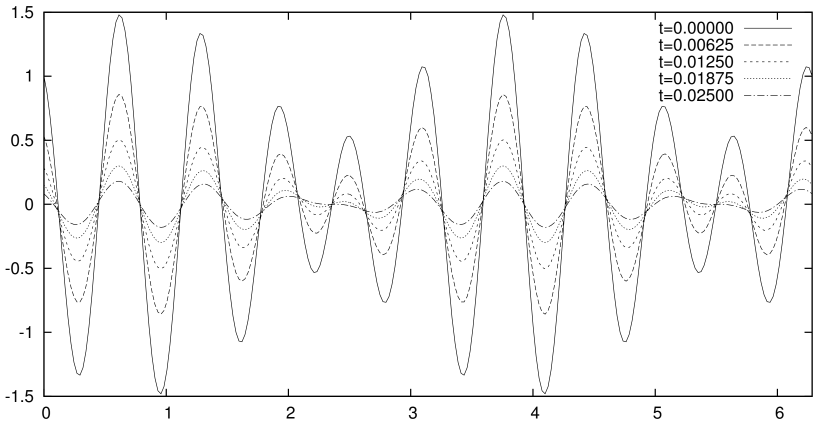

Figure 7.1: Solutions of the diffusion equation (7.2) at different points in time.

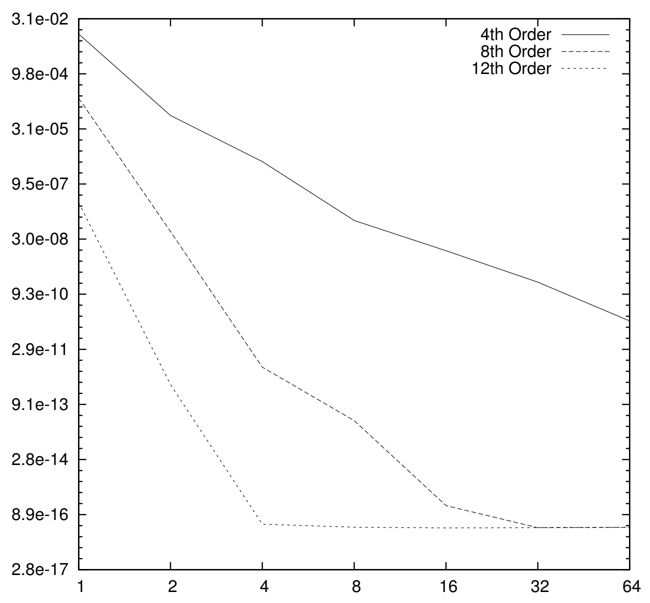

Figure 7.2: Comparison of the L^2-error in computing the solution of the diffusion equation (7.2) for different order methods as a function of the number of subintervals in the temporal domain.

7.2 Dirichlet Boundary Conditions with Constant Coefficients in $\mathbb{R}^1$

We compute the solution for the diffusion equation with constant coefficients

$$\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2} \;\; x \in [0, \pi] \tag{7.6}$$

with initial condition

$$\phi(x, 0) = \sin(12x) - 0.5 \sin(8x) \tag{7.7}$$

and boundary conditions

$$\phi(0, t) = \phi(2\pi, t) = 0 \tag{7.8}$$

In this example, we set $\alpha = 1$.

In this case, the exact solution is given by

$$\phi(x, t) = \sin(12x)e^{-144t} - 0.5 \sin(8x)e^{-64t}. \tag{7.9}$$

Figure 7.3 is a plot of the solution $\varphi(x, t)$ in space for different points in time. In Figure 7.4 we plot the decay of the L^2-error of the solution at $t = 0.1$ as the number of subintervals in the temporal domain increases. We compare schemes of order 4, 8, and 12. In each case, the number of nodes in time m is equal to the order of the scheme and the number of deferred corrections L is equal to $m - 1$. In our spatial discretization, we construct the second-order differentiation matrix with 16 subintervals and 16 points per subinterval. From Figure 5.2, this matrix approximates the exact second derivative of the initial condition in (7.7) with accuracy of about 11 digits.

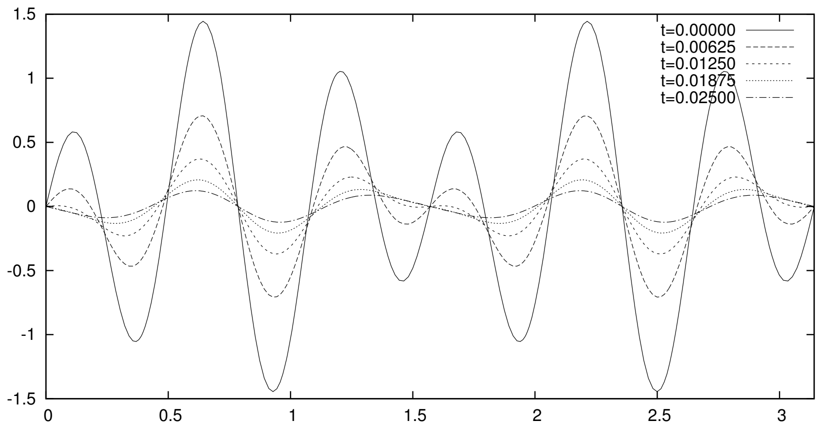

Figure 7.3: Solutions of the diffusion equation (7.6) at different points in time.

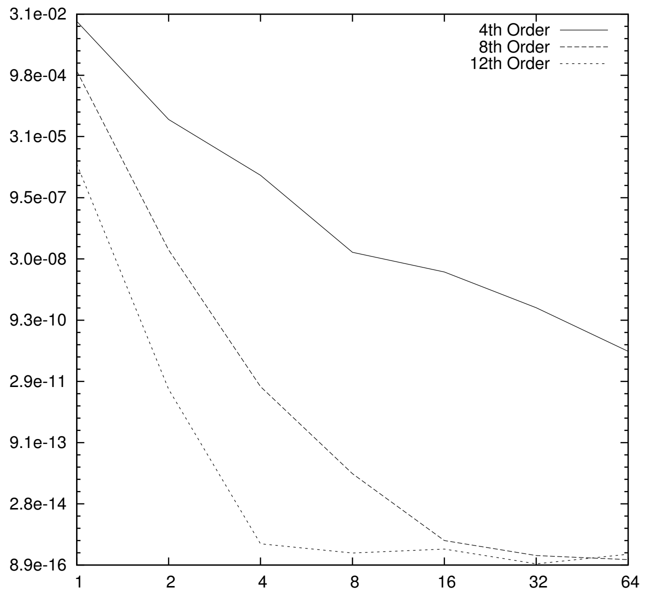

Figure 7.4: Comparison of the L^2-error in computing the solution of the diffusion equation (7.6) for different order methods as a function of the number of subintervals in the temporal domain.

7.3 Dirichlet Boundary Conditions with Variable Coefficients in $\mathbb{R}^2$

We compute the solution for the diffusion equation with variable coefficients

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\alpha(x,y)\nabla\phi) \quad x, y \in [0,\pi] \tag{7.10}$$

with initial condition

$$\phi(x,y,0) = \sin(10x)\,(\cos(2y) - \cos(4y)) \tag{7.11}$$

and homogeneous Dirichlet boundary conditions (i.e., if Γ is the boundary of the square domain $[0,\pi]^2$, then $\varphi(x,y,t) = 0$ for all $(x,y) \in \Gamma$). In this example, we set

$$\alpha(x,y) = (1.1 + \sin(4x))(1.1 + \cos(8y)). \tag{7.12}$$

Figure 7.5 is a plot of the solution $\phi(x,y,t)$ in space for a fixed $y = \frac{\pi}{2}$ and different points in time. Figure 7.6 is a plot of the solution $\phi(x,y,t)$ in space for a fixed $x = \frac{\pi}{4}$ and different points in time. In Figure 7.7 we plot the decay of the L^2-error of the solution at $t = 0.1$ as the number of subintervals in the temporal domain increases. We compare schemes of order 4, 8, and 12. In each case, the number of nodes in time m is equal to the order of the scheme and the number of deferred corrections L is equal to $m-1$. In this case, the error of a solution computed with a given number of subintervals in time is constructed by comparing that solution to the solution computed with twice as many subintervals.

In our spatial discretization, we construct the second-order differentiation matrix in the x-direction with 16 subintervals and 16 points per subinterval. From Figure 5.2, this matrix approximates the exact second derivative of the initial condition in (7.11) for a fixed y with accuracy of about 11 digits. We construct the second-order differentiation matrix in the y-direction with 8 subintervals and 16 points per subinterval. From Figure 5.4, this matrix approximates the exact second derivative of the initial condition (7.11) for a fixed x with accuracy of about 11 digits.

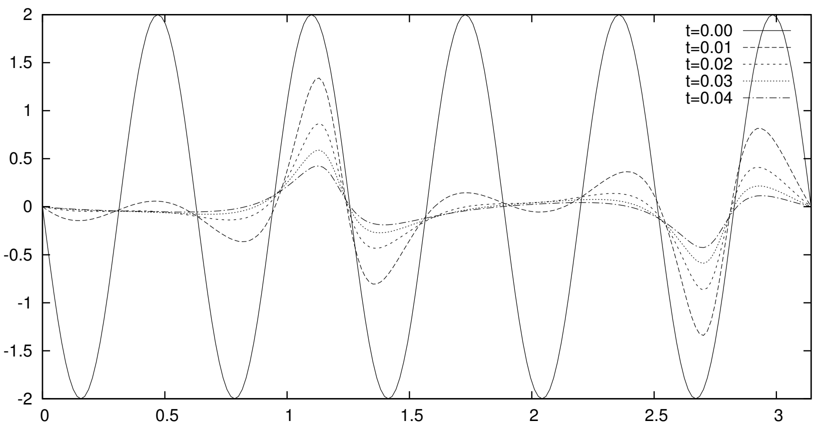

Figure 7.5: Solutions (at $y = \frac{\pi}{2}$) of the diffusion equation (7.10) at different points in time.

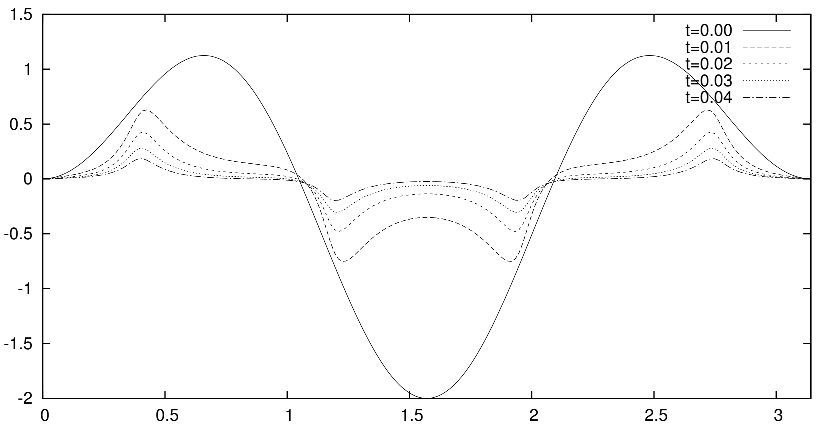

Figure 7.6: Solutions (at $x = \frac{\pi}{4}$) of the diffusion equation (7.10) at different points in time.

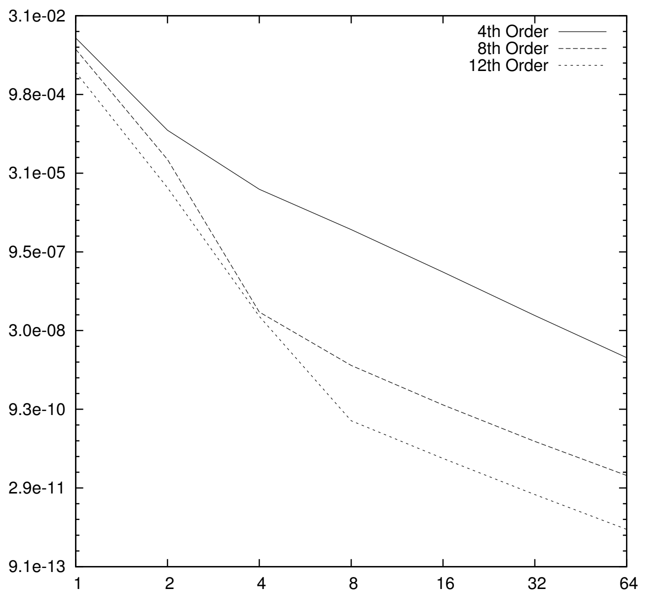

Figure 7.7: Comparison of the L^2-error in computing the solution of the diffusion equation (7.10) for different order methods as a function of the number of subintervals in the temporal domain.

8 Conclusions and Future Work

We introduce a new class of algorithms for the solution of parabolic partial differential equations. We combine Spectral Deferred Correction methods for the solution of systems of ordinary differential equations with the implicit Euler method and Alternating Direction Implicit methods in $\mathbb{R}^2$ and $\mathbb{R}^3$. Furthermore, we construct high-order accurate representations of the spatial operator. We extend the traditional pseudospectral schemes by subdividing the entire spatial domain, constructing bases on each subdomain, and combining the obtained discretization with the implicit SDC schemes. As a result, we construct schemes with arbitrary order of convergence for the solution of parabolic PDEs. Our schemes have CPU time requirements that are linear in the number of spatial discretization nodes and linear in the number of temporal nodes.

Schemes up to order 12 in both time and space have been extensively tested. We are currently in the processes of implementing parallelized and adaptive versions of the approach. An additional advantage of ADI schemes in higher dimensions is that they can be trivially parallelized, which leads to greater computational efficiency. Versions of this approach for the solution of linear and nonlinear problems of wave propagation are being vigorously pursued.

References

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover, 10th ed., 1972.

[2] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi, *Adaptive Solution of Partial Differential Equations in Multiwavelet Bases*, J. Comput. Phys., 182 (2002), pp. 149–190.

[3] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists*, Academic Press, 6th ed., 2005.

[4] K. Bohmer and H. Stetter, eds., *Defect Correction Methods - Theory and Applications*, Springer-Verlag, 1984.

[5] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover, 2nd ed., 2001.

[6] J. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley, 2008.

[7] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, 1988.

[8] G. Dahlquist and Å. Björck, *Numerical Methods in Scientific Computing*, vol. I, Society for Industrial and Applied Mathematics, 2008.

[9] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, 1989.

[10] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral Deferred Correction Methods for Ordinary Differential Equations*, BIT, 40 (2000), pp. 241–266.

[11] A. Dutt and V. Rokhlin, *Fast Fourier Transforms for Nonequispaced Data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393.

[12] L. C. Evans, *Partial Differential Equations*, American Mathematical Society, 2nd ed., 2010.

[13] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1996.

[14] A. Friedman, *Partial Differential Equations of Parabolic Type*, Prentice Hall, 1964.

[15] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[16] D. Gottlieb and L. Lustman, *The spectrum of the chebyshev collocation operator for the heat equation*, SIAM J. Numer. Anal., 20 (1983), pp. 909–921.

[17] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, Society for Industrial and Applied Mathematics, 1977.

[18] I. Gradshteyn and I. Ryzhik, *Table of Integrals, Series, and Products*, Academic Press, 7th ed., 2007.

[19] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM J. Numer. Anal., 28 (1991), pp. 1071–1080.

[20] L. Greengard and P. Lin, *Spectral Approximation of the Free-Space Heat Kernel*, Appl. Comput. Harmonic Anal., 9 (2000), pp. 83–97.

[21] L. Greengard and J. Strain, *A Fast Algorithm for the Evaluation of Heat Potentials*, Comm. Pure Appl. Math., 43 (1990), pp. 949–963.

[22] ——, *The Fast Gauss Transform*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 79–94.

[23] R. Hamming, *Digital Filters*, Courier Dover Publications, 3rd ed., 1989.

[24] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer, 2008.

[25] A. Hoffman, M. Martin, and D. Rose, *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal., 10 (1973), pp. 364–369.

[26] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cam, 2nd ed., 2009.

[27] D. Kushnir and V. Rokhlin, *A highly accurate solver for stiff ordinary differential equations*, SIAM J. Sci. Comput., 34 (2012), pp. A1296–A1315.

[28] A. T. Layton and M. L. Minion, *Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics*, J. Comput. Phys., 194 (2004), pp. 697–715.

[29] ——, *Implications of the Choice of Quadrature Nodes for Picard Integral Deferred Corrections Methods for Ordinary Differential Equations*, BIT, 45 (2005), pp. 341–373.

[30] M. L. Minion, *Semi-implicit projection methods for incompressible flow based on spectral deferred corrections*, Appl. Numer. Math., 48 (2004), pp. 369–387.

[31] ——, *A hybrid parareal spectral deferred corrections method*, Comm. App. Math. and Comp. Sci., 5 (2010), pp. 265–301.

[32] K. Morton and D. Mayers, *Numerical Solution of Partial Differential Equations*, Cambridge University Press, 2nd ed., 2005.

[33] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 3rd ed., 2010.

[34] A. Patera, *A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion*, J. Comput. Phys., 54 (1984), pp. 468–488.

[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 3rd ed., 2007.

[36] J. Reddy, *An Introduction to the Finite Element Method*, McGraw-Hill, 3rd ed., 2005.

[37] B. Riviere, *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*, Society for Industrial and Applied Mathematics, 2008.

[38] K. Sandberg and K. J. Wojciechowski, *The EPS method: A new method for constructing pseudospectral derivative operators*, J. Comput. Phys., 230 (2011), pp. 5836–5863.

[39] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer, 2nd ed., 1993.

[40] J. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*, Springer, 1995.

[41] L. N. Trefethen, *Spectral Methods in MATLAB*, Society for Industrial and Applied Mathematics, 2000.

[42] L. N. Trefethen and M. R. Trummer, *An instability phenomenon in spectral methods*, SIAM J. Numer. Anal., 24 (1987), pp. 1008–1023.

[43] J. Weideman and L. Trefethen, *The eigenvalues of second-order spectral differentiation matrices*, SIAM J. Numer. Anal., 25 (1988), pp. 1279–1298.

[44] N. Yanenko, *The Method of Fractional Steps*, Springer-Verlag, 1971.