NOVEL INTEGRATION IN TIME METHODS VIA DEFERRED CORRECTION FORMULATIONS AND SPACE-TIME PARALLELIZATION

Namdi Brandon

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2015

Approved by:

Jingfang Huang

David Adalsteinsson

Gregory Forest

Laura Miller

Jan Prins

ABSTRACT

Namdi Brandon: Novel Integration in Time Methods via Deferred Correction Formulations and Space-Time Parallelization
(Under the direction of Jingfang Huang)

A major avenue of research in numerical analysis is creating algorithms in order to decrease the amount of computational time in numerical simulations while maintaining high accuracy. Notably when modeling PDE systems, much effort has been focused in creating methods that undergo the spatial calculations very quickly and accurately. Even with these results, simulations may still take too long, limiting the robustness of a numerical model. Hence, a new research direction is to create methods that decrease runtime by focusing on the temporal direction. The subject of this dissertation is the development of algorithms that decrease runtime by taking acount of temporal properties, and when possible coupling both temporal spatial properties, of time-dependent differential equations.

To Anastasia Gage and Dominique Meekers

ACKNOWLEDGEMENTS

First of all, I would like to thank my adviser, Jingfang Huang, for his support and wisdom over these years. I also would like to thank my committee members David Adalsteinsson, Gregory Forest, Laura Miller, and Jan Prins. I would also like to thank Michael Minion and Matthew Emmett for their help and patience in helping me with the beginning of my research. I would like to thank the NSF for providing me financial support during my studies.

I would like to sincerely thank all of my friends that I made during this program. Without their support at work or at play, I could not have made it.

In addition, I would like to extend my gratitude to UNC Chapel Hill and all other schools I have attended for giving me an opportunity to attain an excellent education. And lastly, I would like to thank Anastasia Gage and Dominique Meekers for raising me in an environment that emphasized the value of knowledge.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

0.1 The State of Computing

In 1965, the co-founder of Intel, Gordon Moore, predicted that the transistor density of semiconductor chips, hence the CPU speed, would double roughly every 1.5 years. This prediction has become known as *Moore's law*; and from 1965 to about 2002, Moore's law was upheld [45]. However, as the transistor density increased, the power density of the chip increased causing greater levels of heat on the chip. Technology has reached a point where the speed of processors cannot increase much further due to this limitation. Hence in 2005, a paradigm shift occurred in processor design in hopes of further increasing computational performance. Instead of creating ever-faster CPUs running computations in **serial**, additional performance can be gained by having *multiple* processors work together, or in **parallel** [45].

The ability of having ever increasing computational power and efficient numerical methods that can take advantage of this power has lead to great advances in science. As computational power increases, so does the number of problems previously considered impractical to solve become feasible. Therefore, creating methods to solve these problems is also an increasingly important field of research. The following are a few examples of areas of open problems that require much computational power: climate modeling, data analysis, and molecular dynamics (protein folding and drug discovery).

0.2 Overview of Numerical Methods

For over fifty years, the creation of methods for numerically solving the solutions of time-dependent differential equations has been an active area of research. For ordinary differential equation (ODE) initial value problems (IVPs), various methods such as the linear multistep methods and Runge-Kutta methods have become standard topics in numerical analysis textbooks [1, 2, 28, 40, 49]. In addition, many numerical solvers have become standard tools in order to solve

ODE IVPs such as DASPK, a backward differentiation formula (BDF) based solver [10, 41], and Runge-Kutta method based Radau5 solvers [24]. Numerical solvers have been successfully applied in research and have advanced our knowledge in science and engineering. However, there still presides attributes that limit the effectiveness of existing numerical algorithms. For example, to understand the evolution of charged particles in systems containing thousands of particles, current molecular dynamics simulation tools usually require millions of time steps to accurately capture the motion of particles using existing low order time stepping schemes (e.g., the Verlet integration scheme). Even with the acceleration of the fast N-body solvers [22, 43] for each time step, simulations may require weeks or longer to get physically relevant results.

In recent years, several schemes were introduced to address the challenges in designing accurate and efficient algorithms for large-scale long-time simulations. Examples include the parareal algorithm for parallelization in time [18, 42]; the high order temporal discretization using an orthogonal basis and pseudo-spectral formulations for each time step, to allow larger step sizes [6, 43, 38]; the spectral deferred correction (SDC), integral deferred correction (InDC), iterated defect correction (IDeC), and Krylov deferred correction (KDC) methods for their efficient solutions [3, 14, 16, 30]; and the parallel full approximation scheme in space and time (PFASST), which utilizes parallel computing while combining several preconditioners [17]. The aim of the research presented in this dissertation is to understand the properties of these existing methods and create new methods that take advantage and possibly enhance their favorable traits such as high accuracy, high efficiency, and parallelism.

0.3 Stiff Ordinary Differential Equations

As mentioned earlier, there exists various limitations of numerical methods for time-dependent differential equation systems. The main goal of the research presented in this dissertation is to remedy some of these limitations. One of the aspects known to limit the effectiveness of many numerical algorithms by increasing runtime is stiff ODE IVP systems. In general, a differential equation system is said to be *stiff* if the solution contains signals of multiple time scales: one being smooth and slowly varying (relative to the time interval of the computation) and the others being much more rapidly varying [40]. In other words, Leveque states in [40], "if we perturb the solution

slightly at any time, the resulting curve through the perturbed data has rapid variation. Typically this takes the form of a short lived "transient" response that moves the later solution back toward a smooth solution." To understand stiffness, consider the following ODE

$$y'(t) = \lambda(\, y(t) - \cos(t)\,) - \sin(t). \tag{1}$$

where $\Re(\lambda) < 0$. A solution to this equation is $y(t) = \cos(t)$ with the initial condition $y(0) = 1$. Notice that this smooth solution is the solution for any value of λ. If the initial data is $y(t_0) = \eta$, which does not lie on the curve $\cos(t)$, then the solution through this point is

$$y(t) = e^{\lambda(t-t_0)}(\, \eta - \cos(t_0)\,) + \cos(t) \tag{2}$$

One can verify that this is true through differentiation. Since $\Re(\lambda) < 0$, the function approaches $\cos(t)$ exponentially with decay rate λ. When one perturbs the solution at some point, the perturbed solution approaches the slow changing particular solution $\cos(t)$.

The reason why stiff systems pose hardships on numerical methods for time dependent differential equations is because stiff systems require algorithms to take a much smaller time step in order for an algorithm to be stable. Although the true solution is smooth and it seems that a large time step would suffice, the numerical method must handle the rapidly changing signal by taking smaller time steps in order to attain accuracy. This property of stiff systems increases the runtime of simulations and limits their effectiveness. This documents presents research aimed at overcoming the limitations given by stiff systems.

0.4 Overview of Dissertation

This dissertation will present research on the numerical integration methods used to decrease computational runtime. In chapter 1, we will give an overview of the fast multipole method, the spatial solver we will use when modeling the evolution of charged particles. In chapter 2, we describe and present the properties of the deferred corrections methods, especially the spectral deferred correction method applied to temporal integration. In chapter 3, we present a new integration method based on the Jacobian-free Newton-Krylov method and show initial results when applied

to stiff systems. In chapter 4, we will present the parallel integration method, the parallel full approximation scheme in space and time (PFASST). In chapter 5, we will present an application of PFASST with no spatial-temporal coupling to a geochemical process. In chapter 6, we will present an application of PFASST with spatial-temporal coupling to the evolution of N charged particles. Finally in chapter 7, we will mention future work related to the discussed research.

CHAPTER 1

Fast Multipole Method

In this chapter, we will present a brief overview of the spatial method used for modeling the evolution of charged particles in a vacuum. In this dissertation, we are using the electrostatics assumption, which states that the charge density, denoted by ρ, is stationary or the charge density does not change quickly in time. That is, $\frac{\partial \rho}{\partial t} = 0$ or $\frac{\partial \rho}{\partial t} \ll 1$.

With that being said, the evolution of N charged particles in space is given by Newton's 2^{nd} law. For the i^{th} particle, the equation of motion is

$$\frac{d^2\mathbf{x}_i(t)}{dt^2} = \frac{q_i}{m_i}\mathbf{E}(\mathbf{x}_i(t)) \tag{1.1}$$

where $\mathbf{x}_i$ is the position, $\mathbf{E}_i$ is the electric field, q_i is the charge, and m_i is the mass of the i^{th} particle. Therefore, solving this system consists of two steps. (1) Finding an expression for the electric field. We will designate this as the *spatial calculation*. (2) Once the electric field is calculated, integrating in time to find the new position of the particle. We will designate this as the *temporal calculation*. What follows is an explanation on how to solve for the electric field.

1.1 Poisson Equation

The electrostatic field in a vacuum is described by two of the Maxwell's equations

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \tag{1.2}$$

$$\nabla \times \mathbf{E} = 0 \tag{1.3}$$

where $\rho(x)$ is the charge density within a volume and the constant ϵ_0 is the permeability of free space [32]. Eq. (1.3) is equivalent to expressing $\mathbf{E}$ as the gradient of a scalar function $\Phi(\mathbf{x})$ such that

$$\mathbf{E} = -\nabla\Phi. \tag{1.4}$$

Φ is called the electrostatic potential. Combining Eq. (1.4) and Eq. (1.2), we can write the vector equation for $\mathbf{E}$ in terms of a scalar equation for Φ

$$\nabla^2\Phi = -\frac{\rho}{\epsilon_0}. \tag{1.5}$$

The above equation is called the *Poisson equation*. In regions of space lacking a charge density, Poisson's equation becomes the *Laplace equation*

$$\nabla^2\Phi = 0. \tag{1.6}$$

Hence, solving the spatial calculation for $\mathbf{E}$ in Eq. (1.1) is equivalent to solving the Poisson equation.

1.1.1 Green's Function

The solution to the Poisson equation Eq. (1.5) within a volume V bounded by a surface S can be found by using a construct called a *Green's function*, $G(\mathbf{x}, \mathbf{x}')$. The Green's function for Eq. (1.5) has the property that it is a fundamental solution to the following equation

$$\nabla' G(\mathbf{x}, \mathbf{x}') = -4\pi\delta(\mathbf{x} - \mathbf{x}'). \tag{1.7}$$

Assuming we have found the solution $G(\mathbf{x}, \mathbf{x}')$ to the above equation, the general solution Φ to the Poisson equation is

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\int_V \rho(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')\,\mathrm{d}^3\mathbf{x}' + \frac{1}{4\pi}\oint_S \left[G(\mathbf{x}, \mathbf{x}')\frac{\partial\Phi}{\partial n'} - \Phi(\mathbf{x}')\frac{\partial G(\mathbf{x}, \mathbf{x}')}{\partial n'}\right]\mathrm{d}a' \tag{1.8}$$

where n' is the normal direction pointing out of the surface and $\mathrm{d}a'$ is the area element [32]. Fortunately, there is an explicit formulation of the Green's function for electrostatic problems; it is

$$G(\mathbf{x}, \mathbf{x}') = \frac{1}{|\mathbf{x} - \mathbf{x}'|}. \tag{1.9}$$

1.1.2 Coulomb Potential

Recall that we are interested in finding the potential in free space. The relevant boundary conditions are the Dirichlet conditions. Therefore, we must have that $\Phi(\mathbf{x}) \to 0$ as $\mathbf{x} \to \infty$. In addition, we also have the Green's function satisfy Dirichlet conditions $G|_S = 0$ in Eq. (1.7). Applying the Dirichlet boundary conditions and the appropriate Green's function, Eq. (1.8) becomes

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \int_V \frac{\rho(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} \, \mathrm{d}^3\mathbf{x}'. \tag{1.10}$$

The above equation is called the *Coulomb potential*, and it gives the electrostatic potential subject to the Dirichlet condition in free space for a general charge distribution $\rho(\mathbf{x})$.

By modeling a charged particle as a point charge, we can write the charge density distribution of N charged particles as

$$\rho(\mathbf{x}) = \sum_{i=1}^{N} q_i \delta(\mathbf{x} - \mathbf{x}_i)$$

where q_i is the charge of a particle located at $\mathbf{x}_i$ [32]. Using the above charge distribution in Eq. (1.10) leads to the Coulomb potential for N charged particles

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{j=1}^{N} \frac{q_j}{|\mathbf{x} - \mathbf{x}_j|}. \tag{1.11}$$

We now have all of the components needed to write an explicit formulation for the equations of motion for a system of N charges. Using $\mathbf{E} = -\nabla\Phi$ and Eq. (1.11), we can express Eq. (1.1) as

$$\frac{d^2\mathbf{x}_i(t)}{dt^2} = -\frac{q_i}{m_i} \nabla\Phi_i(\mathbf{x}_i) \tag{1.12}$$

where

$$\Phi_i(\mathbf{x}_i) = \frac{1}{4\pi\epsilon_0} \sum_{j \neq i}^{N} \frac{q_j}{|\mathbf{x}_i - \mathbf{x}_j|}. \tag{1.13}$$

Occasionally in this dissertation, we will make reference to the forces instead of the potential in Eq. (1.12). The force $\mathbb{F}$ is related to the electric field, and hence, the electrostatic potential by $\mathbb{F} = q\mathbf{E} = -q\nabla\Phi$. Using the identity

$$\nabla \left(\frac{1}{|\mathbf{x} - \mathbf{x}'|} \right) = -\frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3},$$

the force of the i^{th} particle is given by

$$\mathbb{F}_i(\mathbf{x}_i) = -q_i \nabla \Phi_i(\mathbf{x}_i) = q_i \sum_{j \neq i}^{N} q_j \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3}. \tag{1.14}$$

1.2 Multipole and Local Expansions

The cost of directly calculating the potentials $\Phi_i(\mathbf{x}_i)$ in Eq. (1.12) for all N particles is $O(N^2)$. When N is large, this cost is too high. In actual applications, we avoid the direct $O(N^2)$ calculation by using approximations. The approximation technique that this dissertation uses is the fast multipole method (FMM), which approximates Eq. (1.13) over all particles with cost $O(N)$ [22, 43].

The FMM takes advantage of the fact that the potential $\Phi(\mathbf{x})$ can be written as a a sum of potentials from two different spatial domains. This property comes from the Green's function

$$G(\mathbf{x}, \mathbf{x}') = \frac{1}{|\mathbf{x} - \mathbf{x}'|}$$

which has spatial multirate properties. We define the following spatial domains Ω_{near} and Ω_{far}, which we will denote as the near-field and the far-field, respectively. For a given location $\mathbf{x}$, a charge found at $\mathbf{x}_i$ has $\mathbf{x}_i \in \Omega_{near}$ with respect to $\mathbf{x}$ if $|\mathbf{x} - \mathbf{x}_i| < R$. And we consider $\mathbf{x}_i \in \Omega_{far}$ with respect to $\mathbf{x}$ if $|\mathbf{x} - \mathbf{x}_i| \geq R$ where R is the characteristic distance that determines the near-field and far-field. In Ω_{far}, the Green's function is smooth; and in Ω_{near}, the Green's function is more singular. Hence, Φ may be written as $\Phi = \Phi_{near} + \Phi_{far}$.

1.2.1 Multipole Expansion

Assuming that $|\mathbf{x}| > |\mathbf{x}'|$, it is convenient to express the Green's function in terms of the series

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \frac{1}{|\mathbf{x}|}\sum_{n=0}^{\infty} P_n(\cos\,\theta)\left(\frac{|\mathbf{x}'|}{|\mathbf{x}|}\right)^n \tag{1.15}$$

where P_n are the Legendre polynomials and θ is the angle between $\mathbf{x}$ and $\mathbf{x}'$. Normally, $\mathbf{x}'$ corresponds to the position of a source charge, so our assumption implies that $\mathbf{x}$ is far from a charge found at $\mathbf{x}'$. This expansion is especially useful in describing far field interactions, since the expansion converges quickly when $\frac{|\mathbf{x}|}{|\mathbf{x}'|} < 1$.

The Legendre polynomial $P_n(u)$ is the solution to the following recursion formulation

$$(2n+1)uP_n(u) = (n+1)P_{n+1}(u) + nP_{n-1}(u)$$

with $P_0(u) = 1$. We can further expand the Legendre polynomials in terms of spherical harmonic functions

$$P_n(\cos\,\theta) = \sum_{m=-n}^{m=n} Y_n^{-m}(\mathbf{x}')Y_n^m(\mathbf{x}) \tag{1.16}$$

to obtain a new formulation of the Green's function in Eq. (1.15)

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{m=n} |\mathbf{x}'|^n Y_n^{-m}(\mathbf{x}')\frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}}. \tag{1.17}$$

If we express $\mathbf{x}$ in spherical coordinates (r,θ,ϕ), the spherical harmonic function $Y_n^m(\mathbf{x})$ is defined as

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(2n+1)}{4\pi}\frac{(n-m)!}{(n+m)!}}\cdot P_n^{|m|}(\cos\,\theta)e^{im\phi} \tag{1.18}$$

where P_n^m are the *associated Legendre polynomials* [32, 52, 31]. The associated Legendre polynomials P_n^m are found by the following formulations

$$\begin{cases} P_n^m(u) & = (1-u^2)^{\frac{m}{2}}\frac{\partial^m}{\partial u^m}P_n(u) \\ P_n^{-m}(u) & = (-1)^m\frac{(n-m)!}{(n+m)!}P_n^m(u). \end{cases}$$

Using the ideas discussed so far, we can express the potential Φ at a position $\mathbf{x} = (r,\theta,\phi)$

expressed in spherical coordinates due to N charged particles at positions $\mathbf{x}_i = (r_i, \theta_i, \phi_i)$ in Eq. (1.11) as

$$\begin{aligned}\Phi(\mathbf{x}) &= \frac{1}{4\pi\epsilon_0}\sum_{i=1}^{N} q_i \sum_{n=0}^{\infty}\sum_{m=-n}^{m=n} |\mathbf{x}_i|^n Y_n^{-m}(\mathbf{x}_i)\frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}} \\ &= \frac{1}{4\pi\epsilon_0}\sum_{n=0}^{\infty}\sum_{m=-n}^{m=n}\sum_{i=1}^{N} q_i|\mathbf{x}_i|^n Y_n^{-m}(\mathbf{x}_i)\frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}}.\end{aligned} \tag{1.19}$$

The terms $\frac{1}{|\mathbf{x}|^{n+1}}$ are called "multipoles," and their coefficients

$$M_n^m = \sum_{i=1}^{N} q_i|\mathbf{x}_i|^n Y_n^{-m}(\mathbf{x}_i) \tag{1.20}$$

are called *moments* of the expansion. Using the the moments M_n^m, we can rewrite the potential as

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\sum_{n=0}^{\infty}\sum_{m=-n}^{m=n} M_n^m\frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}}. \tag{1.21}$$

The above formulation for the potential is called the **multipole expansion**. Notice that if $|\mathbf{x}|$ is large when compared to the charge locations $|\mathbf{x}_i|$, the multipole expansion will converge quickly. Due to this property, we can approximate the infinite sum by truncating the series with p terms

$$\Phi_p(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\sum_{n=0}^{p-1}\sum_{m=-n}^{m=n} M_n^m\frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}}; \tag{1.22}$$

and the residual error of the truncated series is bounded by

$$|\Phi(\mathbf{x}) - \Phi_p(\mathbf{x})| = \frac{Q}{|\mathbf{x}| - |\mathbf{x}_{min}|}\left(\frac{|\mathbf{x}_{max}|}{|\mathbf{x}|}\right)^p \tag{1.23}$$

where $Q = \sum_{i=1}^{N}|q_i|$, $|\mathbf{x}_{min}|$ and $|\mathbf{x}_{max}|$ are the minimum and maximum magnitude of $|\mathbf{x}_i|$, respectively. Note that the amount of calculations needed to calculate $\Phi_p(\mathbf{x})$ is $O(p^2)$.

If we partition space into regions or "boxes" Ω_j of radius R that contain various charges found at $\mathbf{x}_i \in \Omega_j$, the multipole expansion approximation $\Phi_p(\mathbf{x})$ is useful in expressing the potential Φ due to the particles within Ω_j when the point of interest $\mathbf{x}$ is far from region Ω_j. By far, we mean $|\mathbf{x} - \mathbf{u}_j| > R$ where $\mathbf{u}_j$ is the center of region Ω_j. For this reason, $\Phi(\mathbf{x})$ is also known as the "far-field

expansion" with the respect to region Ω_j.

1.2.2 Local Expansion

The multipole expansion approximation assumes that the source particles are in the near field with respect to an origin, $\mathbf{x}_i \in \Omega_{near}$; and the point of interest is in the far field from the origin $\mathbf{x} \in \Omega_{far}$. However, when the case is reversed, $\mathbf{x}_i \, \epsilon \Omega_{far}$ and $\mathbf{x} \, \epsilon \Omega_{near}$ with respect to an origin, we will need a new formulation for the potential Φ. We can use Eq. (1.19) and exchange the vectors $\mathbf{x}$ and $\mathbf{x}_i$ to obtain

$$\begin{aligned} \Phi(\mathbf{x}) &= \frac{1}{4\pi\epsilon_0} \sum_{i=1}^{N} q_i \sum_{n=0}^{\infty} \sum_{m=-n}^{m=n} |\mathbf{x}|^n Y_n^m(\mathbf{x}) \frac{Y_n^{-m}(\mathbf{x}_i)}{|\mathbf{x}_i|^{n+1}} \\ &= \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{\infty} \sum_{m=-n}^{m=n} \sum_{i=1}^{N} q_i \frac{Y_n^{-m}(\mathbf{x}_i)}{|\mathbf{x}_i|^{n+1}} |\mathbf{x}|^n Y_n^m(\mathbf{x}). \end{aligned} \tag{1.24}$$

The above series converges when $\frac{|\mathbf{x}|}{|\mathbf{x}_i|} < 1$. We can express the moments of this expansion L_n^m as

$$L_n^m = \sum_{i=1}^{N} q_i \frac{Y_n^{-m}(\mathbf{x}_i)}{|\mathbf{x}_i|^{n+1}}$$

and rewrite the potential as

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{\infty} \sum_{m=-n}^{m=n} L_n^m \, |\mathbf{x}|^n Y_n^m(\mathbf{x}). \tag{1.25}$$

The above formulation for the potential is called the **local expansion**. Notice that if $|\mathbf{x}|$ is small when compared to the charge locations $|\mathbf{x}_i|$, the local expansion will converge quickly. Due to this property, we can approximate the infinite sum by truncating the series with p terms

$$\Phi_p(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{p-1} \sum_{m=-n}^{m=n} L_n^m \, |\mathbf{x}|^n Y_n^m(\mathbf{x}). \tag{1.26}$$

The local expansion approximation has similar residual error properties as the multipole expansion approximation with the respective values, $|\mathbf{x}|$ and $|\mathbf{x}_i|$ exchanged.

1.3 The $O(N)$ Algorithm

The FMM works by using a tree structure to partition space into boxes that contain certain numbers of particles [22]. For each box m, we define the following spatial domains Ω_{near} and Ω_{far}, which we will denote as the near-field and the far-field, respectively. For a given location $\mathbf{x}$, a charge found at $\mathbf{x}_i$ has $\mathbf{x}_i \in \Omega_{near}$ with respect to $\mathbf{x}$ if $|\mathbf{x} - \mathbf{x}_i| < R$. And we consider $\mathbf{x}_i \in \Omega_{far}$ with respect to $\mathbf{x}$ if $|\mathbf{x} - \mathbf{x}_i| \geq R$ where R is the characteristic distance that determines the near-field and far-field.

In short, the FMM approximates Eq. (1.11) by directly calculating potential due to particles in the near-field and approximating the potential due to particles in the far-field. We can express the FMM approximation $\Psi(\mathbf{x})$ such that $\Psi(\mathbf{x}) \approx \Phi(\mathbf{x})$ as

$$\Psi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\left(\sum_{x_i \epsilon\, \Omega_{near}} \frac{q_i}{|\mathbf{x} - \mathbf{x}_i|} + \sum_{j=0}^{p-1} a_j |\mathbf{x} - \mathbf{x}_c|^j\right). \tag{1.27}$$

We will call Eq. (1.27) the **fast multipole approximation**. In the expansion, $\mathbf{x}_c$ is the position of the box center on the finest level containing $\mathbf{x}$; and a_j are coefficients that depend on $\forall \mathbf{x}_i \in \Omega_{far}$. p is the number of terms in the expansion, and it controls the accuracy of $\Psi(x)$. The larger p is, the more accurate the FMM becomes. The first summation in $\Psi(\mathbf{x})$ is the direct calculation of the potential due to the near-field charges. The second summation in $\Psi(\mathbf{x})$ is the approximation of the potential due to the far-field. We will give a brief explanation of the FMM; but for more information on the FMM, the reader is encouraged to read [22, 31].

For an arbitrary distribution of particles, the FMM uses a hierarchical oct-tree so that each particle is associated with a box at different levels. Each box i has a "parent box" on the next-coarse level to which the i^{th} box is a subset. A box i is a "child box" of box j if i is on the j's subsequent fine level and i is a subset of j. A divide-and-conquer strategy is used to account the far-field interactions of each box on each level by accumulating multipole expansions. Afterwards, the local expansion of a parent box receives the far-field contributions and transmits it to its children [52].

The fast multipole method needs the following properties to approximate the $O(N^2)$ calculation in Eq. (1.13) in $O(N)$ [31].

1. The FMM needs a way to combine several fine-grid multipole expansions into a single coarse-

grid expansion.

2. The FMM needs a way to combine several multipole expansions into a single local expansion about origin of a target box on the same level.

3. The FMM needs a way to translate a box's local expansion to an origin within each of the child boxes at the following fine level of the tree.

1.3.1 Translation of the Multipole Expansion

The following expansion allows us to combine several multipole expansions of one level into a single expansion on a coarser level. If we have a multipole expansion about the origin, we can shift the expansion and center it at a point $\mathbf{z}$ [31]. The original expansion about the origin

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{\infty} \sum_{m=-n}^{m=n} M_n^m \frac{Y_n^m(\mathbf{x})}{|\mathbf{x}|^{n+1}}$$

can be written as an expansion about $\mathbf{z}$ as

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{\infty} \sum_{m=-n}^{m=n} \tilde{M}_n^m \frac{Y_n^m(\mathbf{x}-\mathbf{z})}{|\mathbf{x}-\mathbf{z}|^{n+1}}$$

where

$$\tilde{M}_n^m = \sum_{j=0}^{n} \sum_{k=-j}^{j} M_{n-j}^{m-k} \frac{i^{|m|}}{i^{|k|} i^{|m-k|}} \frac{A_j^k A_{n-j}^{m-k}}{A_n^m} |\mathbf{z}|^j Y_j^k(\mathbf{z})$$

and the constant A_n^m is defined by

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}.$$

The error bounds of the truncated expansion is

$$|\Phi(\mathbf{x}) - \Phi_p(\mathbf{x})| \leq \frac{Q}{|\mathbf{x}-\mathbf{z}| - |\mathbf{x}_i| - |\mathbf{z}|} \left(\frac{|\mathbf{x}_i| + |\mathbf{z}|}{|\mathbf{x}-\mathbf{z}|} \right)^p.$$

1.3.2 Conversion to a Local Expansion

The following shows how to convert a multipole expansion to a local expansion on the same level. We must assume that the new center at $-\mathbf{z}$ must be far enough away from the multipole expansion assumed at the origin such that $|\mathbf{z}| > (1+c)|\mathbf{x}|$.

$$\Phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} L_n^m |\mathbf{z}-\mathbf{x}| Y_n^m(\mathbf{z}-\mathbf{x})$$

where

$$L_n^m = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{(-1)^j} \frac{i^{|m-k|}}{i^{|k|} i^{|m|}} \frac{A_j^k A_n^m}{A_{j-n}^{k-m}} \frac{Y_{j+n}^{k-m}(\mathbf{z})}{|\mathbf{z}|^{j+n-1}}$$

and A_j^k is defined as before. The error bounds of a truncated expansion with p terms, Φ_p is

$$|\Phi(\mathbf{x}) - \Phi_p(\mathbf{x})| \leq \frac{Q}{(c-1)|\mathbf{x}|} \left(\frac{1}{c}\right)^p.$$

1.3.3 Translation of the Local Expansion

To shift a local expansion of a box on the parent level to a box center of the child box at $-\mathbf{z}$, we start with the local expansion given by

$$\Phi_p(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{p-1} \sum_{m=-n}^{m=n} L_n^m \, |\mathbf{x}|^n Y_n^m(\mathbf{x}).$$

We can express the truncated expansion

$$\Phi_p(\mathbf{x}) = \frac{1}{4\pi\epsilon_0} \sum_{n=0}^{p-1} \sum_{m=-n}^{m=n} \tilde{L}_n^m \, |\mathbf{x}-\mathbf{z}|^n Y_n^m(\mathbf{x}-\mathbf{z})$$

where

$$\tilde{L}_n^m = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{L_j^k}{(-1)^{j+n}} \frac{i^{|k|}}{i^{|k-m|} i^{|m|}} \frac{A_{j-n}^{k-m} A_n^m}{A_j^k} Y_{j-n}^{k-m}(-\mathbf{z}) |\mathbf{z}|^{j-n}$$

where A_k^j is defined as before [31].

1.3.4 The Algorithm Outline

The FMM consists of the following steps [31]:

1. Form multipole expansions (moments) at the finest scale.
2. Merge (translate) expansions to form expansions on the next coarser level until the coarsest scale is reached.
3. Starting at the coarsest level, for each target region, convert the multipole expansion into local expansion at the center of each target box.
4. For each box, merge (translate) the local expansion to the center of each of a box's children until the finest level is reached.
5. Add the near-field potential contribution from the nearest neighbors to the approximated far-field potentials to obtain Eq. (1.27).

1.4 Multirate FMM

The FMM is an extremely useful method that takes advantage of the spatial properties of the Coulomb potential $\Phi(\mathbf{x}(t))$. However, we can take advantage of the temporal properties of the Coulomb potential and fast multipole approximation $\Psi(\mathbf{x}(t))$ to make a less computationally expensive algorithm.

From Eq. (1.14), one can see that the magnitude of the forces is proportional to $\frac{1}{r^2}$ where $r = |\mathbf{x}_j - \mathbf{x}_i|$. When $\mathbf{x}_j \in \Omega_{near}$ with respect to $\mathbf{x}_i$, we can expect that the near-field forces should change rapidly in time due to the $\frac{1}{r^2}$ force with r small. When $\mathbf{x}_j \in \Omega_{far}$ with respect to $\mathbf{x}_i$, we can expect that the far-field forces should change slowly in time due to the $\frac{1}{r^2}$ force with r large. Thus, we can represent the forces in Eq. (1.1) as

$$\frac{d^2\mathbf{x}_i}{dt^2} = -\frac{q_i}{m_i}\nabla\left(\Phi_{\text{near}}(\mathbf{x}_i) + \Phi_{\text{far}}(\mathbf{x}_i)\right). \tag{1.28}$$

This can be rewritten as

$$\frac{d^2\mathbf{x}_i}{dt^2} = -\frac{q_i}{4\pi\epsilon_0 m_i}\nabla\left(\sum_{x_j\in\Omega_{near}}\frac{q_j}{|\mathbf{x}_j-\mathbf{x}_i|}+\sum_{x_j\in\Omega_{far}}\frac{q_j}{|\mathbf{x}_j-\mathbf{x}_i|}\right).$$

This formulation suggests that the forces have a dual behavior as they change in time, reminiscent of a stiff system. The total force has two different time scales: a fast changing near-field and a slow changing far-field. Since the FMM approximation Ψ, defined in Eq. (1.27), approximates the right hand side of Eq. (1.28), we should expect Ψ to uphold this multirate behavior. More importantly, it should be possible to take account of this multirate behavior in time integration schemes. We will call the process of exploiting the temporal multirate behavior of the FMM as the **multirate FMM** (MRFMM).

1.4.1 Numerical Evidence

To test our hypothesis of the inherent temporal multirate behavior of the FMM, we ran numerical experiments to see how the far-field and near-field potentials change in time. To do this, we simulated the the motion of particles while keeping track of $\Psi_{near}(\mathbf{x}(t))$ and $\Psi_{far}(\mathbf{x}(t))$ for each time step. Once the simulation is over, for a given particle, we calculated the respective least-squares polynomial that approximates $\Psi_{near}(\mathbf{x}(t))$ and $\Psi_{far}(\mathbf{x}(t))$ over time. Afterwards for each time step, we calculated the L_∞ norm of the error between $\Psi_{near}(\mathbf{x}(t))$ and $\Psi_{far}(\mathbf{x}(t))$ and their respective least-squares approximating polynomial over all particles.

For a fixed L_∞ error, the slower a potential varies in time (ie. a smoother potential), the lower the degree of the least-squares polynomial is needed. Our intuition says that for a fixed error between the numerical FMM solution and the least-squares polynomial, the least squares polynomial corresponding to $\Psi_{far}(\mathbf{x})$ should have a lower degree than that of $\Psi_{near}(\mathbf{x})$. The results of the experiment show this to be true. That is, for a fixed error, there is a difference in scale between $\Psi_{far}(\mathbf{x})$ and $\Psi_{near}(\mathbf{x})$. One needs a least-squares polynomial of lower degree for $\Psi_{far}(\mathbf{x})$ than that of $\Psi_{near}(\mathbf{x})$.

Our experimental setup was as follows. We first designated two types of particles: sources and targets. Source particles had randomly distributed charges in $[-\frac{1}{2},\frac{1}{2}]$. Source particles were used to provide the electrostatic potential, and they were able to move in space. The target particles had

charge 1. They simply moved in space but did not have any effect on the potential due to neither the source particles nor due the other target particles. The potential at the all locations were due solely to the source particles. We also assumed that all particles have mass equal to 1.

The numerical experiment was done using 16,000 source particles and 16,000 target particles randomly distributed in a unit cube. We used the FMM approximation $\Psi(\mathbf{x})$ to approximate the Coulomb potential $\Phi(\mathbf{x})$ such that the error tolerance in $\Psi(\mathbf{x})$ was 0.5×10^{-9}. During the simulation, the FMM tree was fixed as well. The simulation consisted of 200 time steps with size $\Delta t = 10^{-7}$ so that $t_{start} = 0$ and $t_{final} = 2 \times 10^{-5}$. And the time-marching scheme that we used was the velocity Verlet method, which is as follows

$$\begin{aligned} \mathbf{x}(t_{n+1}) &= \mathbf{x}(t_n) + \mathbf{v}(t_n)\Delta t + \frac{\mathbf{a}(t_n)}{2}\Delta t^2 \\ \mathbf{v}(t_{n+1}) &= \mathbf{v}(t_n) + \frac{\Delta t}{2}(\mathbf{a}(t_{n+1}) + \mathbf{a}(t_n)) \end{aligned} \tag{1.29}$$

where $\Delta t = t_{n+1} - t_n$; $\mathbf{x}(t_n)$, $\mathbf{v}(t_n)$, and $\mathbf{a}(t_n)$ are the position, velocity, and acceleration, respectively at time $t = t_n$ [26]. The velocity Verlet method has many favorable properties. Namely, it is explicit, $O(\Delta t^2)$, and symplectic [26]. For future reference, we will call running this experiment as the *multirate test*.

Figure 1.1 shows the results of our experiment. We plot the L_∞ error over all target particles between $\Psi_{far}(\mathbf{x})$ and $\Psi_{near}(\mathbf{x})$ and their respective least squares approximating polynomials for various degrees. We represent the error logarithmically, showing the digits of precision in the error.

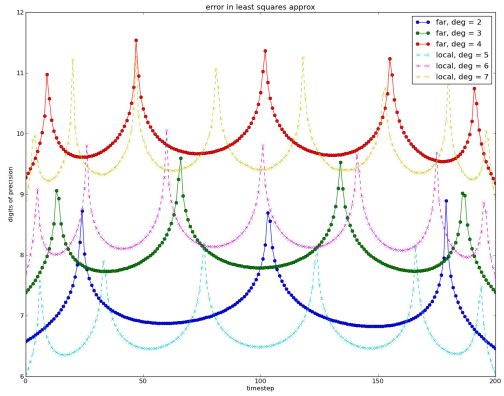


Figure 1.1: Results of the multirate test

These results provide numerical evidence of the inherit temporal multirate behavior in the FMM. We can see that for the same error, there is about a 3-degree separation in the least squares polynomial between the smoother $\Psi_{far}(\mathbf{x})$ than $\Psi_{near}(\mathbf{x})$. That is for a fixed error, $\Psi_{far}(\mathbf{x})$ needs a lower degree least-squares polynomial than that for Ψ_{near}. This agrees with our intuition; the far-field potential should change more slowly in time than the far-field potential. Thus, we have two distinct temporal behaviors. We will take advantage of the FMM's inherent multirate behavior in what we will call the MRFMM in modeling Eq. (1.1).

CHAPTER 2

Deferred Correction Methods

It is desirable for algorithms to efficiently converge to solutions of large-scale long-time simulations for ODEs with high accuracy. One way to obtain such high accuracy solutions is to create methods that converge to high order temporal collocation formulations. Solvers that have been made to directly calculate accurate collocation formulation solutions have shown to not be able to do so efficiently. For example in [23, 25], the Gauss collocation formulations using only 2, 4, and 6 nodes were implemented as geometric integrators for Hamiltonian systems. Unfortunately, numerical results show that without the aid of deferred correction or other acceleration techniques, these solvers may not be able to calculate a highly accurate solution as efficiently as other linear multistep methods (see Fig. 5.1 in [23]). If an algorithm can obtain high accuracy at the cost of severely reduced efficiency, the algorithm becomes impractical in application. Hence, it is of great importance to create algorithms that obtain high accuracy while at the same time obtaining high efficiency.

What follows is a perspective of understanding and integrating methods in a numerical framework for solving ODE systems by calculating high order collocation formulations (highly accurate solutions). In this framework, we consider the deferred correction techniques as efficient iterative schemes to reduce the error in the **convergence procedure**; and different deferred correction strategies can be applied to reduce different error components in the provisional solution. Within the prescribed convergence tolerance, we will analyze the mathematical properties of the solution by studying the underlying collocation formulations. In the optimal numerical implementation of this framework, the collocation formulation is selected based on the physical properties of the solution. We treat each low order deferred correction scheme as a preconditioner, and integrate these preconditioning techniques with existing iterative solvers (e.g., fixed point iterations or Jacobian-free Newton-Krylov methods) for better convergence. By understanding the different properties of various collocation formulations for high accuracy and the different convergence procedures for high efficiency, we will lay the ground work for creating an "optimal" algorithm for attaining high temporal accuracy.

This chapter is organized as follows. In Sec. 2.1, we study the **converged solution** by developing the "collocation formulations database" for the numerical framework for solving ODE initial value problems and by discussing the properties of each formulation. In Sec. 2.2, we start from the backward Euler based spectral deferred correction methods and their convergence properties, and then study different deferred correct methods to form the "deferred correction methods database" in the **convergence procedure**, an iterative procedure to reduce the errors in the provisional solution. In Sec. 2.4, we discuss several algorithm design guidelines to integrate different components to **efficiently converge** to the solution of an **"optimal" discretization** in the numerical framework.

2.1 Collocation Formulations and Properties

For long time simulations, it is in general impractical to use one single step for the entire interval from $t = 0$ to t_{final} (e.g., by using a spectral formulation for $[0, t_{final}]$). What is done in practice is that the entire time interval is divided into a sequence of subintervals (time steps) based on the properties of the solution and any step size constraints. In this section, we discuss different collocation formulations for each time step. These collocation formulations differ in the mathematical formulations, choices of collocation points, and numerical integration or differentiation strategies. We leave the discussions of their accurate and efficient solutions to later sections.

Spectral and pseudo-spectral methods have been widely used for solving spatial differential equations in simple geometries (i.e., Fourier series for periodic solutions, or Chebyshev polynomials for rectangular or cubic geometries) [11, 20, 21]. One advantage of these methods is that when the number of expansion terms (in the spectral formulation) or node points (in the pseudo-spectral type collocation formulation) increases, the approximation error decays very rapidly for smooth functions. And unlike traditional linear multistep methods or low order explicit Runge-Kutta methods for the temporal initial value problems, the stability region constraint is in general not a big concern. Not surprisingly, as time is only one dimensional and there is no complex geometry involved, these methods have also been applied for solving time-dependent differential equations in the past. In this section, we first discuss the Legendre polynomial based Gauss collocation formulation, and then discuss other collocation formulations for initial value problems. We would like to emphasize that when an iterative scheme is applied to a specific collocation formulation and is convergent

(up to a prescribed precision), the numerical properties of the solution are then determined by the properties of the collocation formulation, not the convergence procedure. Unlike existing analysis of the deferred correction methods, this new viewpoint allows us to study the mathematical properties of the framework (e.g., order and stability) by focusing on the converged solution of the collocation formulation, and to consider the *convergence procedure* (describing how the iterations converge) separately.

2.1.1 Gauss Collocation Method

We first present a variant of the well-studied Gauss collocation formulation (also referred to as the Gauss Runge-Kutta (GRK) method) for ODE initial value problems $y'(t) = f(t, y(t))$ with given initial data $y(0)$ [25, 28]. To march one step from $t = 0$ to $t = \Delta t$, we define $Y(t) = y'(t)$ as the new unknown function and recover $y(t)$ using $y(t) = y(0) + \int_0^t Y(\tau)d\tau$. This will give what we call the **"yp-formulation"** as

$$Y(t) = f(t, y(0) + \int_0^t Y(\tau)d\tau). \tag{2.1}$$

In the Gauss collocation formulation, p *Gaussian quadrature nodes* $\mathbf{t} = [t_1, t_2, \cdots, t_p]^T$ are used to discretize the yp-formulation in $[0, \Delta t]$. For the given function values $\mathbf{Y} = [Y_1, Y_2, \cdots, Y_p]^T$ at the Gaussian nodes, we can construct the $(p-1)^{th}$ degree Legendre polynomial expansion to approximate $Y(t) = y'(t)$ where the coefficients are computed using the Gaussian quadrature rules. We can integrate this interpolating polynomial analytically from 0 to t_m, where $1 \leq m \leq p$, to form a linear mapping that maps the function values $\mathbf{Y}$ to the integral of $Y(t)$ at the node points. Taking out the scalar factor Δt in this mapping, the integral $\int_0^t Y(\tau)d\tau$ can be approximated by $\Delta t S\mathbf{Y}$, where S is called the **"spectral integration matrix"** [21] which can be precomputed. The discretized Gauss collocation formulation using p node points in the time interval $[0, \Delta t]$ is given by

$$\mathbf{Y} = \mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t S\mathbf{Y}). \tag{2.2}$$

The following theorem, mostly from [28], summarizes several nice properties of this formulation, assuming it is solved exactly.

Theorem 2.1.1. *For ODE initial value problems, the Gauss collocation formulation in Eq. (2.2) with p nodes is of order 2p (super convergence), A-stable, B-stable, symplectic (structure preserving), and symmetric (time reversible). In addition, the error decays exponentially when p increases.*

Interested readers are referred to [5, 27] for the proof of the theorem. These nice properties allow the use of very large time step sizes when solving ordinary differential equation initial value problems.

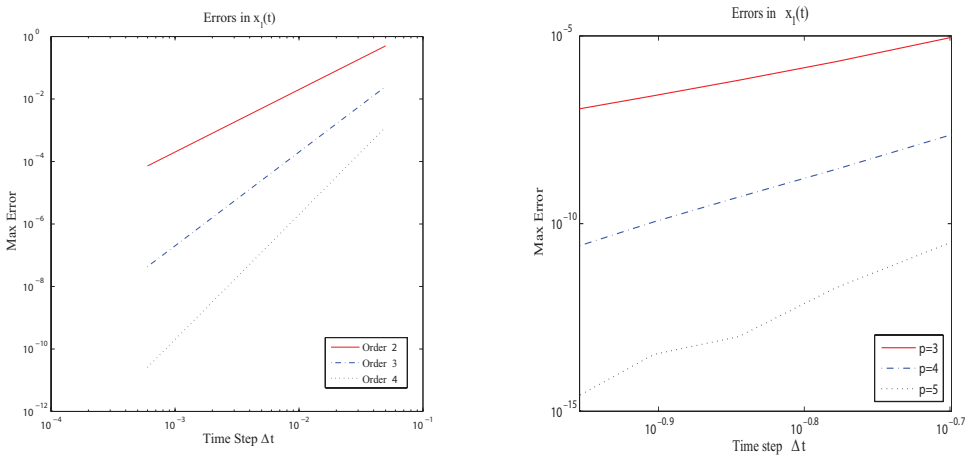
Comment: The yp-formulation can be easily generalized to differential algebraic equations (DAEs) of the form $F(t, y, y') = 0$, and the discretized system becomes

$$\mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t S\mathbf{Y}, \mathbf{Y}) = \mathbf{0}.$$

Similar to the ODE case, the pseudo-spectral type collocation formulation allows much larger time step sizes in the numerical simulation. In Fig. 2.1, we compare the Gauss collocation formulation with traditional BDF methods for the DAE system from [1]

$$\begin{pmatrix} x_1' \\ x_2' \\ 0 \end{pmatrix} = \begin{pmatrix} 10 - \frac{1}{2-t} & 0 & 10(2-t) \\ \frac{9}{2-t} & -1 & 9 \\ t+2 & t^2-4 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ z \end{pmatrix} + \begin{pmatrix} \frac{3-t}{2-t}e^t \\ 2e^t \\ e^t(2-t-t^2) \end{pmatrix} \tag{2.3}$$

whose analytical solution is given by $(x_1, x_2, z) = \left(e^t, e^t, -e^t/(2-t)\right)$ and can be resolved to machine precision using a 15-term Legendre polynomial expansion for each component when $t \in [0, 1]$. Due to the stiffness of the DAE, the fourth order BDF method requires a time step size of 10^{-3} for 10 digits of accuracy, as shown in (a) of Fig. 2.1 (also see [1], p.268). On the other hand, the Gauss collocation discretization using a step size of 10^{-1} and 5 Gaussian nodes gives 14 digits accuracy (see (b) in Fig. 2.1). Detailed analysis of different collocation formulations for DAE systems can be found in [24] and references therein. More examples demonstrating the step size-accuracy relations of the pseudo-spectral type collocation formulations for both ODE and DAE problems can be found in [29, 30].



(a) (b)

Figure 2.1: Accuracy in x_1 for different step sizes using (a) traditional BDF methods, orders 2, 3, 4 (from [1]) and (b) Gauss collocation methods using 3, 4, 5 Gaussian nodes.

2.1.2 Different Collocation Formulations

In the Gauss collocation formulation discussed in the previous section, the Legendre polynomial based Gaussian quadrature nodes are used and the spectral integration matrix is constructed accordingly for the yp-formulation. Other types of formulations, quadrature nodes, and numerical differentiation or integration techniques have also been studied in the literature. In this subsection, we present different collocation formulations to form our "collocation formulation database" for ODE initial value problems.

For the ODE initial value problem $y' = f(t, y)$, most existing collocation formulations use y as the unknown and solve the differential equation directly. In the "differential quadrature method" [13] and other traditional pseudo-spectral collocation formulations, the "spectral differentiation matrix" is constructed by differentiating the interpolating polynomial of y at the collocation points and evaluating the derivative polynomial to form the spectral differentiation matrix D mapping $\mathbf{y}$ at the collocation points to $\mathbf{y}'$. We refer to this class of formulations as the **"differential formulation,"** and the discretized ODE system can be represented as $D\mathbf{y} = \mathbf{f}(\mathbf{t}, \mathbf{y})$. An alternative formulation is to use the equivalent Picard integral equation formulation $y(t) = y_0 + \int_0^t f(\tau, y(\tau))d\tau$ and discretize the ODE system as in

$$\mathbf{y} = \mathbf{y}_0 + \Delta t S \mathbf{F}(\mathbf{t}, \mathbf{y}) \tag{2.4}$$

where $\mathbf{y}$ are the unknowns at the collocation points, and S is the (scaled) spectral integration matrix. We refer to this formulation as the **"integral formulation."** It is important to note that when the spectral integration matrix is used, this integral formulation also converges to the Gauss collocation formulation. When this formulation is coupled with uniform collocation points, the resulting deferred correction methods are called the integral deferred correction methods (InDC) [14]. In the previous subsection, we also presented the **"yp-formulation"** using y' as the unknown and using the spectral integration matrix to form the discretized collocation formulation given by $\mathbf{Y} = \mathbf{f}(\mathbf{t}, \mathbf{y}_0 + \Delta t S \mathbf{Y})$. Although these formulations are equivalent mathematically, they have very different numerical properties. Also, it is not easy to generalize some of these formulations to more complicated differential equation systems. For example, for a general DAE system $F(t, y, y') = 0$, it is nontrivial to derive the standard Picard integral equation for y in the integral formulation, and one may prefer the differential formulation or yp-formulation. On the other hand, we will see later that an integral formulation approach may be better suited for handling stiff systems than a yp-formulation approach.

Instead of Gaussian quadrature nodes, other node points have also been studied in the literature: when **Radau Ia** nodes are used, the left end-point $t = 0$ is added when constructing the numerical integration or differentiation matrices; when **Radau IIa** nodes are used, the right end-point $t = \Delta t$ is added. In the **Gauss-Lobatto** scheme, both end points are added in the collocation formulation. One can also use the Chebyshev polynomial based **Clenshaw-Kurtis quadrature** and the corresponding spectral differentiation or integration matrices to take advantage of the "near-minimax" approximation properties of the Chebyshev polynomial expansion and the fast Fourier transform [50]. Since these collocation points are closely related with the underlying orthogonal polynomials, one can very stably construct the least squares polynomial using the corresponding Gaussian-type quadratures and differentiate or integrate the resulting polynomial to construct the spectral differentiation or integration matrices. Note that for ODE problems, when considering the errors at both the interior and boundary collocation points, these collocation formulations have similar order properties as shown in traditional ODE analysis. However, when only considering the solution at the right end point $t = \Delta t$, the Legendre polynomial based collocation formulations are preferred due to their relatively higher order of convergence. Also, for DAE problems, the orders at $t = \Delta t$ will be different for the "differential" and "algebraic" components (see, e.g. [28]) for

different choices of nodes; and the Radau IIa or Gauss-Lobatto nodes are usually preferred due to their relative higher order properties for the algebraic components.

More recently, assuming the solution can be better approximated by **exponential sums** as in the case for linear homogeneous ODEs, collocation nodes and spectral integration matrices are designed using skeletonization techniques by Rokhlin et. al. for ODE systems [19, 38]. When the solution can be approximated by the so-called "band-limited" functions, in [6], quadrature nodes and the corresponding spectral integration matrix using the **"prolate spheroidal wave functions"** were applied to initial value problems. These collocation formulations only differ in the set of node points and precomputed spectral differentiation matrix D or integration matrix S. It is therefore possible to precompute and form a collocation formulation database. For a given ODE system, based on the physical properties of the solution and different measures of the error, one can choose a particular set of nodes and the corresponding matrix to form the "optimal" formulation. Also note that unlike traditional ODE solvers, for better accuracy, in addition to changing to a smaller step size and reducing the error using the "order of convergence" concept, one can also add more points to the interval in the collocation formulation to take full advantage of the convergence properties in the orthogonal basis based pseudo-spectral methods. The latter option may be more favorable if the resulting system can be solved efficiently, and usually allows much larger step sizes in the simulation.

When a smaller number of nodes (e.g., less than 10 node points) is preferred (e.g., due to memory constraints), in the existing integral deferred correction methods [14], the uniform nodes are usually applied as they show better convergence properties in the deferred correction iterations as will be discussed in the next section. However, such uniform collocation formulations may have serious numerical problems (especially when the number of nodes increases) due to the stability and accuracy issues from the underlying uniform polynomial interpolation schemes, such as the well-known Runge's phenomenon. We believe such collocation formulations should be avoided in the final converged solution; however, one may want to take advantage of their fast convergence in the deferred correction iterations as will be discussed in Section 2.2. Also, generalization of the collocation schemes to partial differential equations is straightforward and interested readers are referred to [12, 33, 34] for preliminary results along this direction.

2.2 Deferred Correction Methods and Properties

Despite the aforementioned excellent properties of many of the high order collocation formulations, the higher order ($p \geq 10$ node points) collocation formulation is rarely used in most of today's numerical simulations. The main reason is the efficiency of the solution algorithms. Assuming an ODE system with N equations is resolved using p Gaussian nodes in the Gauss collocation formulation, as the spectral differentiation matrix D or integration matrix S is dense (solutions at current time depend both on history data and solutions at future times), the Newton's method and direct Gauss elimination (for each linearized system) will require $O((Np)^3)$ operations. This number increases cubicly as p increases. In most BDF type methods, the operation is only N^3 for each time step. Also, when the step size is large, the initial value may no longer serve as a good initial guess for the solution in the time interval, resulting in convergence problems in the nonlinear solver.

Instead of direct Gauss elimination, in recent years, different deferred correction methods were proposed to improve the efficiency when solving the discretized collocation formulations iteratively. We first present the backward Euler based spectral deferred correction (SDC) methods for the yp-Gauss collocation formulation.

2.2.1 Backward Euler Preconditioned SDC for yp-Gauss Collocation Formulation

We consider the yp-formulation in Eq. (2.2) using the Gaussian nodes. The first step in a SDC method is to use a low order "predictor" to find an approximate solution of $Y(t)$ at the collocation points in $[0, \Delta t]$, denoted by $\tilde{\mathbf{Y}} = [\tilde{Y}_1, \tilde{Y}_2, \cdots \tilde{Y}_p]^T$. When the backward Euler's method is applied, the predictor solves the low order discretized system given by

$$\begin{aligned}
\tilde{Y}_1 &= f(t_1, y_0 + \Delta t_1 \tilde{Y}_1) \\
\tilde{Y}_2 &= f(t_2, y_0 + \Delta t_1 \tilde{Y}_1 + \Delta t_2 \tilde{Y}_2) \\
\vdots &\quad \cdots \\
\tilde{Y}_p &= f(t_p, y_0 + \Delta t_1 \tilde{Y}_1 + \Delta t_2 \tilde{Y}_2 \cdots + \Delta t_p \tilde{Y}_p)
\end{aligned}$$

where $\Delta t_i = t_i - t_{i-1}$ ($t_0 = 0$) is the time step size from t_{i-1} to t_i. In matrix form, this is equivalent to solving

$$\tilde{\mathbf{Y}} = \mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t \tilde{S} \tilde{\mathbf{Y}}) \tag{2.5}$$

where

$$\Delta t \tilde{S} = \begin{bmatrix} \Delta t_1 & 0 & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & 0 & 0 \\ \vdots & & & & \\ \Delta t_1 & \Delta t_2 & \cdots & \Delta t_{p-1} & \Delta t_p \end{bmatrix} \tag{2.6}$$

is the first order rectangular rule (using the right end point) for approximating $\int_0^{t_i} Y(\tau) d\tau$. Unlike the spectral integration matrix S where solutions at current time depend on both the history and future data, in the low order discretization represented succinctly in Eq. (2.5), solutions are "decoupled" due to the lower triangular structure of $\tilde{S}$. This reduces the solution time to $O(N^3 p)$ for ODE systems of size N, assuming Gauss elimination is used for each step of the Newton iterations when solving the nonlinear system $\tilde{Y}_k = f(t_k, y_0 + \Delta t_1 \tilde{Y}_1 + \Delta t_2 \tilde{Y}_2 \cdots + \Delta t_k \tilde{Y}_k)$ when marching from t_{k-1} to t_k. We use $\tilde{Y}(t)$ to represent the corresponding Legendre interpolating polynomial of $\tilde{\mathbf{Y}}$, where the expansion coefficients are stably computed using the Gaussian quadrature.

In the second step of the SDC method, we define the error as $\delta(t) = Y(t) - \tilde{Y}(t)$. We can express the **"error's equation"** given by

$$\tilde{Y}(t) + \delta(t) = f(t, y_0 + \int_0^t (\tilde{Y}(\tau) + \delta(\tau)) d\tau) \tag{2.7}$$

with initial value $\delta(0) = 0$. Since $\tilde{Y}$ at the Gaussian nodes is given, we can apply the spectral integration matrix to $\int_0^t \tilde{Y}(\tau) d\tau$ to accurately evaluate the integral. For the unknown $\delta(t)$, similar to the predictor step, the backward Euler's method can be applied to obtain a low order approximation of the error $\delta(t)$ by solving the equation system

$$\tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}} = \mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}) \tag{2.8}$$

where $\tilde{\boldsymbol{\delta}} = [\tilde{\delta}_1, \tilde{\delta}_2, \cdots, \tilde{\delta}_p]^T$ is the low order solution at each collocation node. Next, we can add $\tilde{\boldsymbol{\delta}}$

to $\tilde{\mathbf{Y}}$ to obtain an "improved" approximation of $Y(t)$, define the new error, and repeat the second step. We refer to each such iteration as one SDC correction. In the SDC methods, this procedure is stopped either when $\tilde{\boldsymbol{\delta}}$ is smaller than a prescribed accuracy requirement or after a fixed number of iterations. In the latter case, if the error is still large, one reduces the step size and solves the collocation formulation in a smaller interval. In other words, one accepts the SDC results only when $\tilde{\boldsymbol{\delta}}$ in Eq. (2.8) is within certain error tolerance. Notice that in this case, $\tilde{\mathbf{Y}}$ approximately satisfies (up to $O(\tilde{\boldsymbol{\delta}})$ error)

$$\tilde{\mathbf{Y}} = \mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t S \tilde{\mathbf{Y}}) \tag{2.9}$$

which is exactly the Gauss collocation formulation in Eq. (2.2). Therefore, SDC is simply an iterative scheme trying to converge to the Gauss collocation formulation.

Comment: When analyzing the deferred correction methods, most existing results follow traditional numerical ODE theory and study the convergence and stability region properties for varying step size Δt. However, note that when the error is large in the deferred correction iterations, the results will not be accepted and smaller step sizes have to be used until the error is small enough. This implies that most existing analyses cover inapplicable numerical regimes which never appear in real implementations. It is therefore more appropriate to separate the study of the *convergence procedure* from that of the converged solutions. When the corrections are convergent, the numerical properties of the algorithm are determined by the underlying collocation formulation.

Comment: Generalization of the SDC methods to the DAE problems is straightforward. When the backward Euler's method is applied, the corresponding low order discretization for the error is given by $\mathbf{F}(\mathbf{t}, \mathbf{y}_0 + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}) = \mathbf{0}$. For a given provisional solution, only $O(N^3 p)$ operations are required to get the low order error approximation $\tilde{\boldsymbol{\delta}}$ in each SDC correction due to the lower triangular structure of $\tilde{S}$.

2.2.2 Backward Euler Preconditioned SDC for integral-Gauss Collocation Formulation

We now consider the integral formulation in Eq. (2.4) using the Gaussian nodes and present a similar analysis as the previous subsection. As mentioned earlier, SDC begins with a predictor.

And applying the backward Euler's method, we obtain the provisional solution

$$\begin{aligned}
\tilde{y}_1 &= y_0 + \Delta t_1 f(t_1, \tilde{y}_1) \\
\tilde{y}_2 &= y_0 + \Delta t_1 f(t_1, \tilde{y}_1) + \Delta t_2 f(t_2, \tilde{y}_2) \\
\vdots &\quad \cdots \\
\tilde{y}_p &= y_0 + \Delta t_1 f(t_1, \tilde{y}_1) + \Delta t_2 f(t_2, \tilde{y}_2) + \cdots + \Delta t_p f(t_p, \tilde{y}_p)
\end{aligned}$$

where $\Delta t_i = t_i - t_{i-1}$ ($t_0 = 0$) is the time step size from t_{i-1} to t_i. In matrix form, this is equivalent to solving

$$\tilde{\mathbf{y}} = \mathbf{y_0} + \Delta t \tilde{S} \mathbf{F}(\mathbf{t}, \tilde{\mathbf{y}}) \tag{2.10}$$

where $\Delta t \tilde{S}$ is the same as in Eq. (2.6).

In the second step of the SDC method, we define the error as $\delta(t) = y(t) - \tilde{y}(t)$. The error's equation is now given by

$$\tilde{y}(t) + \delta(t) = y_0 + \int_0^t f(\tau, \tilde{y}(\tau) + \delta(\tau)) \mathrm{d}\tau \tag{2.11}$$

with initial value $\delta(0) = 0$. We then obtain a low order approximation of the error $\delta(t)$ by solving the equation system

$$\tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}} = \mathbf{y}_0 + \Delta t S \mathbf{F}(\mathbf{t}, \tilde{\mathbf{y}}) + \Delta t \tilde{S} (\mathbf{F}(\mathbf{t}, \tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}}) - \mathbf{F}(\mathbf{t}, \tilde{\mathbf{y}})) \tag{2.12}$$

where $\tilde{\boldsymbol{\delta}}$ is the low order solution at each collocation node. Next, we continue the SDC procedure by adding $\tilde{\boldsymbol{\delta}}$ to $\tilde{\mathbf{y}}$ to obtain an improved approximation to $y(t)$, defining a new error, and repeating the second step. When $\tilde{\boldsymbol{\delta}}$ is within a certain error tolerance in Eq. (2.12), the approximation $\tilde{\mathbf{y}}$ approximately satisfies (up to $O(\tilde{\boldsymbol{\delta}})$ error)

$$\tilde{\mathbf{y}} = \mathbf{y}_0 + \Delta t S \mathbf{F}(\mathbf{t}, \tilde{\mathbf{y}}) \tag{2.13}$$

which is the Gaussian collocation formulation in Eq. (2.4).

2.2.3 Understanding Deferred Correction Iterations

To gain further insight of the deferred correction iterations, we first consider the SDC scheme in matrix form applied to a linear ODE of the form $y'(t) = \lambda y + f(t)$ with given initial condition $y(0) = y_0$. We will present an analysis using both the yp-formulation and the integral formulation.

The corresponding collocation formulations for the yp-formulation and integral formulation, respectively become

$$\mathbf{Y} = \lambda(\mathbf{y}_0 + \Delta t S\mathbf{Y}) + \mathbf{F} \tag{2.14}$$

$$\mathbf{y} = \mathbf{y}_0 + \Delta t S(\lambda\mathbf{y} + \mathbf{F}) \tag{2.15}$$

where $\mathbf{y}_0 = [y_0, y_0, \cdots, y_0]^T$ and $\mathbf{F} = [f(t_1), f(t_2), \cdots, f(t_p)]^T$. Therefore the linear systems for $\mathbf{Y}$ and $\mathbf{y}$ are given by

$$(I - \lambda\Delta t S)\mathbf{Y} = \lambda\mathbf{y}_0 + \mathbf{F} \tag{2.16}$$

$$(I - \lambda\Delta t S)\mathbf{y} = \mathbf{y}_0 + \Delta t S\mathbf{F}. \tag{2.17}$$

In the first step of SDC, using the backward Euler's method as the predictor to solve the low order discretization for the respective formulations

$$(I - \lambda\Delta t \tilde{S})\mathbf{Y} = \lambda\mathbf{y}_0 + \mathbf{F} \tag{2.18}$$

$$(I - \lambda\Delta t \tilde{S})\mathbf{y} = \mathbf{y}_0 + \Delta t S\mathbf{F}, \tag{2.19}$$

we obtain the respective initial provisional solutions

$$\mathbf{Y}^{[0]} = (I - \lambda\Delta t \tilde{S})^{-1}(\lambda\mathbf{y}_0 + \mathbf{F}) \tag{2.20}$$

$$\mathbf{y}^{[0]} = (I - \lambda\Delta t \tilde{S})^{-1}(\mathbf{y}_0 + \Delta t S\mathbf{F}). \tag{2.21}$$

Assuming the provisional solution from the previous SDC correction is denoted by $\mathbf{Y}^{[n]}$ and $\mathbf{y}^{[n]}$, the discretized low order error's equations in Eq. (2.8) and Eq. (2.12) for the respective formulations

become

$$\mathbf{Y}^{[n]} + \tilde{\boldsymbol{\delta}} = \lambda(\mathbf{y}_0 + \Delta t S \mathbf{Y}^{[n]} + \Delta t \tilde{S}\tilde{\boldsymbol{\delta}}) + \mathbf{F} \tag{2.22}$$

$$\mathbf{y}^{[n]} + \tilde{\boldsymbol{\delta}} = \mathbf{y}_0 + \lambda(\Delta t S \mathbf{Y}^{[n]} + \Delta t \tilde{S}\tilde{\boldsymbol{\delta}}) + \Delta t S \mathbf{F}. \tag{2.23}$$

For the yp-formulation, using Eq. (2.20) to write $(\lambda \mathbf{y}_0 + \mathbf{F})$ as $(I - \lambda \Delta t \tilde{S})\mathbf{Y}^{[0]}$, $\tilde{\boldsymbol{\delta}}$ is then given by

$$\tilde{\boldsymbol{\delta}} = \mathbf{Y}^{[0]} - (I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t S)\mathbf{Y}^{[n]}. \tag{2.24}$$

Similarly for the integral formulation, using Eq. (2.20) to write $\mathbf{y}_0 + \Delta t S \mathbf{F}$ as $(I - \lambda \Delta t \tilde{S})\mathbf{y}^{[0]}$, the integral formulation correction is then given by

$$\tilde{\boldsymbol{\delta}} = \mathbf{y}^{[0]} - (I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t S)\mathbf{y}^{[n]}. \tag{2.25}$$

Notice that the correction equations for the yp-formulation and the integral formulation mirror each other. Therefore, we have the same recursive relation for both formulations

$$\mathbf{Y}^{[n+1]} = \mathbf{Y}^{[n]} + \tilde{\boldsymbol{\delta}} = \mathbf{Y}^{[0]} + C\mathbf{Y}^{[n]} \tag{2.26}$$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \tilde{\boldsymbol{\delta}} = \mathbf{y}^{[0]} + C\mathbf{y}^{[n]} \tag{2.27}$$

where the matrix C is given by

$$\begin{aligned} C &= I - (I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t S) \\ &= I - (I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t \tilde{S} + \lambda \Delta t \tilde{S} - \lambda \Delta t S) \\ &= (I - \lambda \Delta t \tilde{S})^{-1} \lambda \Delta t (S - \tilde{S}), \end{aligned}$$

and is what we call the **"correction matrix"** in this dissertation. We would like to emphasize the fact that the correction matrix is the same for both the yp and integral formulation based SDC schemes. Solving the recursive equation in Eq. (2.26) for the yp-formulation and Eq. (2.27) for the

integral formulation, we obtain

$$\mathbf{Y}^{[n]} = \mathbf{Y}^{[0]} + C\mathbf{Y}^{[0]} + C^2\mathbf{Y}^{[0]} + \cdots + C^n\mathbf{Y}^{[0]} \tag{2.28}$$

$$\mathbf{y}^{[n]} = \mathbf{y}^{[0]} + C\mathbf{y}^{[0]} + C^2\mathbf{y}^{[0]} + \cdots + C^n\mathbf{y}^{[0]}. \tag{2.29}$$

Instead of the above step-by-step analysis of the SDC method, a more straightforward viewpoint is to consider the collocation formulations in Eq. (2.16) and Eq. (2.17) and apply the low-order preconditioner $(I - \lambda\Delta t\tilde{S})^{-1}$ to get the preconditioned systems

$$(I - \lambda\Delta t\tilde{S})^{-1}(I - \lambda\Delta tS)\mathbf{Y} = (I - \lambda\Delta t\tilde{S})^{-1}(\lambda\mathbf{y}_0 + \mathbf{F}) = \mathbf{Y}^{[0]} \tag{2.30}$$

$$(I - \lambda\Delta t\tilde{S})^{-1}(I - \lambda\Delta tS)\mathbf{y} = (I - \lambda\Delta t\tilde{S})^{-1}(\mathbf{y}_0 + \Delta tS\mathbf{F}) = \mathbf{y}^{[0]}. \tag{2.31}$$

Since $\tilde{S}$ is a low order approximation of S (or when $\lambda\Delta t$ is small), $(I-\lambda\Delta t\tilde{S})^{-1}(I-\lambda\Delta tS) = I-C$ is close to the identity matrix. Applying the Neumann series to the equation $(I-C)\mathbf{Y} = \mathbf{Y}^{[0]}$, we can derive Eq. (2.28) directly. Likewise, applying the Neumann series to the equation $(I-C)\mathbf{y} = \mathbf{y}^{[0]}$, we can derive Eq. (2.29) directly. Therefore, for **linear ODE problems**, we conclude that the SDC method is simply a **Neumann series expansion** for solving the optimal collocation formulation preconditioned by the low order methods. The convergence of the deferred correction methods is then determined by the following theorem.

Theorem 2.2.1. *For linear ODE initial value problems, the spectral deferred correction iterations in Eq. (2.28) are convergent if and only if the* **spectral radius** $\rho(C)$ *(the supremum among the absolute values of all the eigenvalues) of the correction matrix* C *is less than* 1.

For nonlinear problems, the SDC approach can be considered as a simplified Newton's method for the yp-formulation. For a given input provisional solution $\mathbf{Y}^{[k]}$, denoting the low order approximation of the error $\tilde{\delta}$ as an implicit function of $\mathbf{Y}^{[k]}$ as $\tilde{\delta} = \mathbf{H}(\mathbf{Y}^{[k]})$, one can apply the Newton's method to find the zero of $\mathbf{H}$,

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} - J_H^{-1}\mathbf{H}(\mathbf{Y}^{[k]}) = \mathbf{Y}^{[k]} - J_H^{-1}\tilde{\delta}.$$

To find an expression for $J_H^{-1} = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}}$, we can start with the general error equation for DAEs

$$\mathbf{0} = \mathbf{F}(\mathbf{t}, \tilde{\mathbf{u}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}})$$

where $\tilde{\mathbf{u}} = \mathbf{y}_0 + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}$. As in [30], we use the implicit function theorem to obtain s

$$\begin{aligned}
\mathbf{0} &= \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \left(\Delta t S + \Delta t \tilde{S} \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} \right) + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} \left(I + \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} \right) \\
\implies \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t \tilde{S} \right) \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} &= - \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t S \right) \\
\implies \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} &= - \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t \tilde{S} \right)^{-1} \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t S \right) \\
\frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} &= - \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t \tilde{S} \right)^{-1} \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t S - \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t \tilde{S} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t S \right) \\
J_H = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{Y}}} &= -I + \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} + \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t \tilde{S} \right)^{-1} \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{u}}} \Delta t (S - \tilde{S}) \right).
\end{aligned}$$

Here, we can think of the Jacobian matrix as $J_H = -I + C$ (note that $\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{Y}}} = I$ for ODE systems). The Jacobian matrix is close to the negative Identity matrix $-I$ when the low-order preconditioner is effective, since $\tilde{S}$ is an approximation to S and Δt is small. Therefore, the Newton's method in

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} - J_H^{-1} \mathbf{H}(\mathbf{Y}^{\tilde{[\mathbf{k}]}}) = \mathbf{Y}^{[k]} - J_H^{-1} \tilde{\delta}$$

is simplified to $\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} + \tilde{\delta}$.

We can show that SDC is a simplified Newton's method for the integral formulation as well. For a given input provisional solution $\mathbf{y}^{[k]}$, denoting the low-order approximation of the error $\tilde{\delta}$ as an implicit function of $\mathbf{y}^{[k]}$ as $\tilde{\delta} = \mathbf{H}(\mathbf{y}^{\tilde{[\mathbf{k}]}})$, one can apply the Newton's method to find the zero of $\mathbf{H}$,

$$\mathbf{y}^{[k+1]} = \mathbf{y}^{[k]} - J_H^{-1} \mathbf{H}(\mathbf{y}^{\tilde{[\mathbf{k}]}}) = \mathbf{y}^{[k]} - J_H^{-1} \tilde{\delta}.$$

To find an expression for $J_H^{-1} = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{y}}}$, we can start with the error equation for the integral formulation

in Eq. (2.12) and use the implicit function theorem to obtain

$$
\begin{aligned}
I + \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{y}}} &= \Delta t S \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} + \Delta t \tilde{S} \left(\frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} (\tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}}) \left(I + \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{y}}} \right) - \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} \right) \\
\implies \left(I - \Delta t \tilde{S} \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} (\tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}}) \right) \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{y}}} &= -I + \Delta t \tilde{S} \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} (\tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}}) + \Delta t (S - \tilde{S}) \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} \\
\implies J_H = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \tilde{\mathbf{y}}} &= -I + \left(I - \Delta t \tilde{S} \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} (\tilde{\mathbf{y}} + \tilde{\boldsymbol{\delta}}) \right)^{-1} \left(\Delta t (S - \tilde{S}) \frac{\partial \mathbf{F}}{\partial \tilde{\mathbf{y}}} \right).
\end{aligned}
$$

Once again, we can think of the Jacobian matrix as $J_H = -I + C$. Thus, the Newton's method in

$$\mathbf{y}^{[k+1]} = \mathbf{y}^{[k]} - J_H^{-1} \mathbf{H}(\mathbf{y}^{\tilde{[\mathbf{k}]}}) = \mathbf{y}^{[k]} - J_H^{-1} \tilde{\delta}$$

is simplified to $\mathbf{y}^{[k+1]} = \mathbf{y}^{[k]} + \tilde{\delta}$.

2.2.4 Properties of Deferred Correction Iterations

Our numerical results (also see [16]) show that for many ODE initial value problems, the properly implemented deferred correction methods outperform many existing commonly used solvers in efficiency for the same accuracy requirement, especially when very high accuracy (i.e., more than 6 digits accuracy) is required. However, we also observe the "order reduction" phenomenon when deferred correction iterations are applied to very stiff ODE systems. For some DAE systems, the deferred correction scheme becomes divergent, independent of the selected step size. We refer interested readers to Fig. 7 in [30], where the SDC method is applied to Andrews' squeezing problem (see [44] for the full description of this DAE system) and becomes divergent after a few iterations for different step sizes. One observation is that when the Gauss collocation formulation is solved exactly, "order reduction" or divergence is never a concern in the converged solution. This observation means that the order reduction or divergence is not caused by the final converged solution, but by the deferred correction *convergence procedure*, in particular, the spectral radius $\rho(C)$ of the correction matrix C and the error in the **initial provisional solution**.

We first define the "convergence region" to measure when the deferred correction methods are convergent for linear problems.

Definition 2.2.2. For linear ODE initial value problems, we define the "convergence region" Ω

of a deferred correction method as $\Omega = \{\lambda\Delta t : \ \rho(C(\lambda\Delta t)) < 1, \ \lambda \in \mathbb{C}\}$. The method is called "A-convergent" if Ω contains the left half complex plane. It is called "L-convergent" if it is "A-convergent" and $\lim_{|\lambda\Delta t|\to\infty} \rho(C(\lambda\Delta t)) \to 0$ for $\lambda\Delta t$ on the left half complex plane.

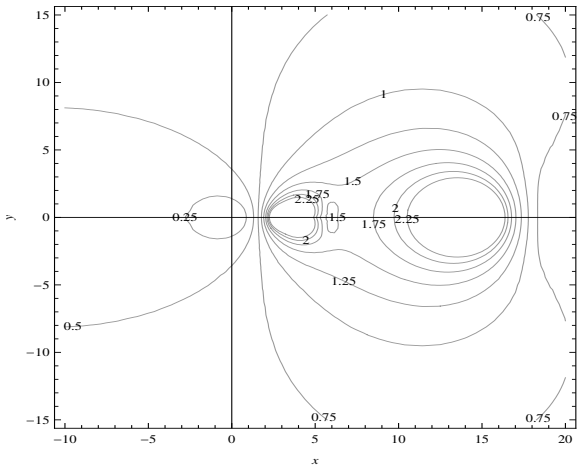


Figure 2.2: Contour of $\rho(C(\lambda\Delta t))$ for $p = 4$ for *SDC*, $\lambda\Delta t = x + iy$.

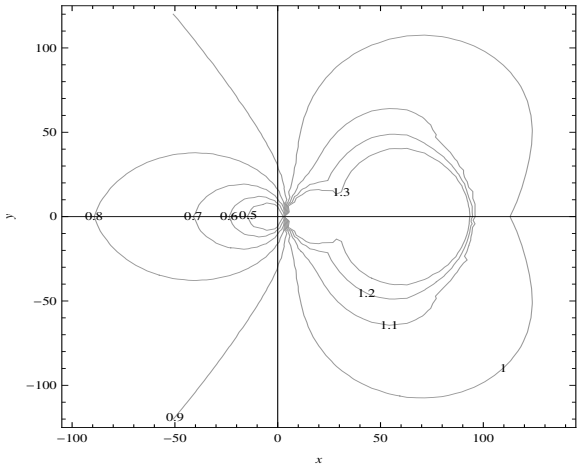


Figure 2.3: Contour of $\rho(C(\lambda\Delta t))$ for $p = 10$ for *SDC*, $\lambda\Delta t = x + iy$.

For the backward Euler preconditioned SDC methods for yp-Gauss collocation formulation, the correction matrix is

$$C = (I - \lambda\Delta t\tilde{S})^{-1}(\lambda\Delta t)(S - \tilde{S}). \tag{2.32}$$

In Figs. 2.2 and 2.3, we plot the numerically computed convergence region (contour = 1) and other contour lines of $\rho(C)$ for $p = 4$ and $p = 10$. Both seem to be A-convergent.

For the correction matrix $C(\lambda\Delta t)$, we are particularly interested in two regimes to understand

the properties of the deferred correction iterations: when $|\lambda\Delta t| \ll 1$ (non-stiff systems), and when $|\lambda\Delta t| \to \infty$ ("strongly stiff limit" for stiff systems). For **non-stiff systems** where $|\lambda\Delta t| \ll 1$, after each iteration, clearly the error will decay approximately by the factor $(\lambda\Delta t)(S - \tilde{S})$ as

$$\begin{aligned} C_{\rm ns} &= (I + (\lambda\Delta t\tilde{S}) + (\lambda\Delta t\tilde{S})^2 + \cdots)(\lambda\Delta t)(S - \tilde{S}) \quad (2.33) \\ &= (\lambda\Delta t)(I + (\lambda\Delta t\tilde{S}) + (\lambda\Delta t\tilde{S})^2 + \cdots)(S - \tilde{S}). \quad (2.34) \end{aligned}$$

However in the **strongly stiff limit**, the correction matrix becomes

$$\begin{aligned} C_{\rm s} &= -(\lambda\Delta t\tilde{S})^{-1}(\lambda\Delta t)(S - \tilde{S}) \\ C_{\rm s} &= I - \tilde{S}^{-1}S. \quad (2.35) \end{aligned}$$

The convergence of the iterations will then depend on how accurate the low order integration rule in $\tilde{S}$ approximates the high order rule in S. In Table 2.1, we list $\rho(C_{\rm s})$ for different numbers of node points. It can be seen that "order reduction" becomes a serious problem as the number of nodes

Table 2.1: $\rho(C_{\rm s})$ for different numbers of Gaussian nodes, stiff case, *SDC*.

p	2	3	4	5	6	7	8
$\rho(C_{\rm s})$	0.3170	0.4210	0.5610	0.6653	0.7420	0.7998	0.8448
p	9	10	11	12	13	14	15
$\rho(C_{\rm s})$	0.8805	0.9096	0.9337	0.9540	0.9713	0.9861	0.9991
p	16	17	18	19	20	25	50
$\rho(C_{\rm s})$	1.0105	1.0205	1.0295	1.0375	1.0448	1.0724	1.1280

increases. For 8 points, the modulus of the largest eigenvalue of the correction matrix is 0.8448. This means that for general stiff ODE systems, one error component will decay asymptotically by the factor 0.8448 after each SDC iteration due to the "unresolved" stiff components (as $|\lambda\Delta t| \gg 1$) in the iterations. When $p = 16$, the SDC method becomes divergent as $\rho(C_{\rm s}) = 1.0105$. Clearly, when $p > 15$, the methods are not A-convergent, and the error will eventually start to increase when the number of iterations increases. For several cases when $p \leq 15$, our numerical results show that the methods are A-convergent. Also, from Table 2.1, we see that none of these methods are L-convergent. In Fig. 2.4, we also plot the eigenvalue distributions of $C_{\rm s}$ for $p = 10$ and $p = 40$.

In addition to spectral radius $\rho(C)$ which determines the asymptotic convergence properties of

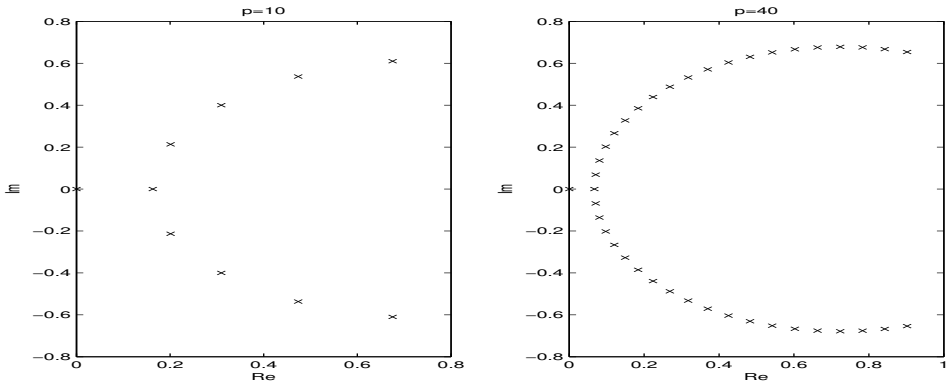
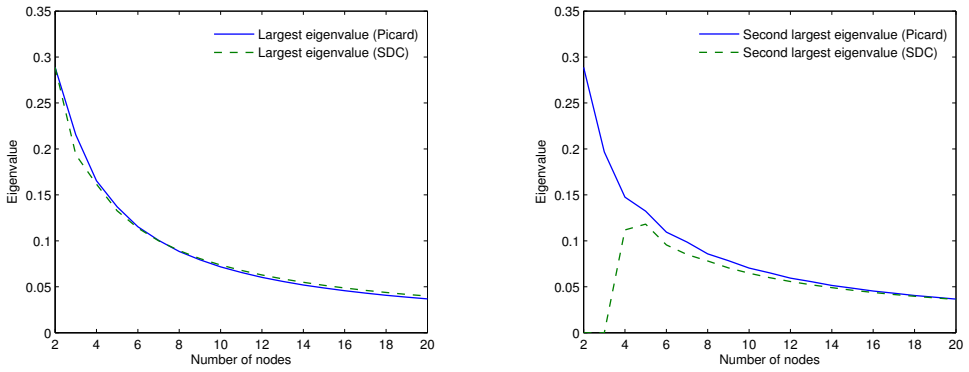


Figure 2.4: Distributions of correction matrix eigenvalues for $p = 10$ and $p = 40$, stiff case, *SDC*.

the deferred correction iterations, the initial error (and its corresponding eigen-decomposition) in the provisional solution also plays an important role in the "convergence procedure". This will be explained in this subsection by comparing the SDC iterations with standard Picard iterations for **non-stiff** linear ODE systems (Picard iterations are divergent for stiff systems).

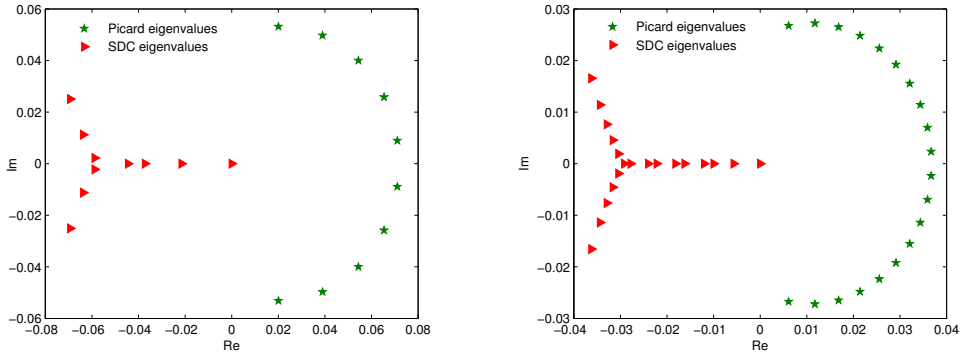


(a) (b)

Figure 2.5: Modulus of the (a) largest and (b) second largest eigenvalues for different numbers of nodes, *SDC vs. Picard for the Gauss collocation formulation.*

In the standard Picard iteration, the solution is derived by applying the Neumann series directly to $(I - \lambda\Delta tS)\mathbf{Y} = b \equiv (\lambda\mathbf{y}_0 + \mathbf{F})$ as $\mathbf{Y} = b + C_{\text{ns}}^{\text{P}}b + (C_{\text{ns}}^{\text{P}})^2b + \cdots$ for the yp-formulation or $(I - \lambda\Delta tS)\mathbf{y} = b \equiv (\mathbf{y}_0 + \Delta tS\mathbf{F})$ as $\mathbf{y} = b + C_{\text{ns}}^{\text{P}}b + (C_{\text{ns}}^{\text{P}})^2b + \cdots$ for the integral formulation. For both cases, the new correction matrix is given by $C_{\text{ns}}^{\text{P}} = \lambda\Delta tS$. To understand the asymptotic convergence properties, we notice that after each Picard iteration, similar to the SDC iterations, the error will be reduced by a factor of $O(\lambda\Delta t)$. We therefore compare the constant prefactor

determined by the spectral radius of $S - \tilde{S}$ in the SDC correction matrices C_{ns} and the radius of S in the Picard correction matrix C_{ns}^{P}. In (a) of Fig. 2.5, we compare the spectral radius (modulus of the largest eigenvalue $|\lambda|_{max}$) of S for Picard iteration and that of $S - \tilde{S}$ for SDC. It can be seen that asymptotically the SDC iterations have a similar convergence rate as the Picard iterations when $\lambda \Delta t$ is small. In (b) of Fig. 2.5, we also show how the second largest eigenvalues change as a function of the number of Gaussian nodes for the SDC and Picard iterations. In Fig. 2.6, we plot the eigenvalue distributions of the matrix $S - \tilde{S}$ in the SDC method and S in the Picard iterations for (a) $p = 10$ and (b) $p = 20$, respectively. In Fig. 2.7, we plot the normalized eigenvectors of the matrix $S - \tilde{S}$, and in Fig. 2.8, the normalized eigenvectors of S, both for $p = 15$. These vectors can be considered as the discretized eigenfunctions. Each component v_j in the eigenvector $\mathbf{v}$ is considered as the eigenfunction value at t_j.



(a) (b)

Figure 2.6: Eigenvalue distributions of SDC and Picard iterations for (a) 10 nodes and (b) 20 nodes

One interesting observation is that even though the spectral radii of the two correction matrices are similar in magnitude (which implies similar convergence rates for a large number of iterations), the eigenvalue distributions and structures of the eigenvectors are very different. To see this behavior, note that we can use Eq. 2.26 and Eq. 2.28 to express the solution of the yp-formulation as

$$\begin{aligned}\mathbf{Y}^{[n]} &= \mathbf{Y}^{[0]} + C\mathbf{Y}^{[0]} + C^2\mathbf{Y}^{[0]} + \cdots + C^n\mathbf{Y}^{[0]} \\ &= \mathbf{Y}^{[0]} + \tilde{\boldsymbol{\delta}}^{[0]} + C\tilde{\boldsymbol{\delta}}^{[0]} + \cdots + C^{n-1}\tilde{\boldsymbol{\delta}}^{[0]}\end{aligned}$$

where C can be the Picard correction matrix C_{ns}^{p} or the non-stiff SDC correction matrix C_{ns} and $\tilde{\boldsymbol{\delta}}^{[0]}$ is the initial error. Expressing $\tilde{\boldsymbol{\delta}}^{[0]}$ in the eigenvector (of the correction matrix) decomposition

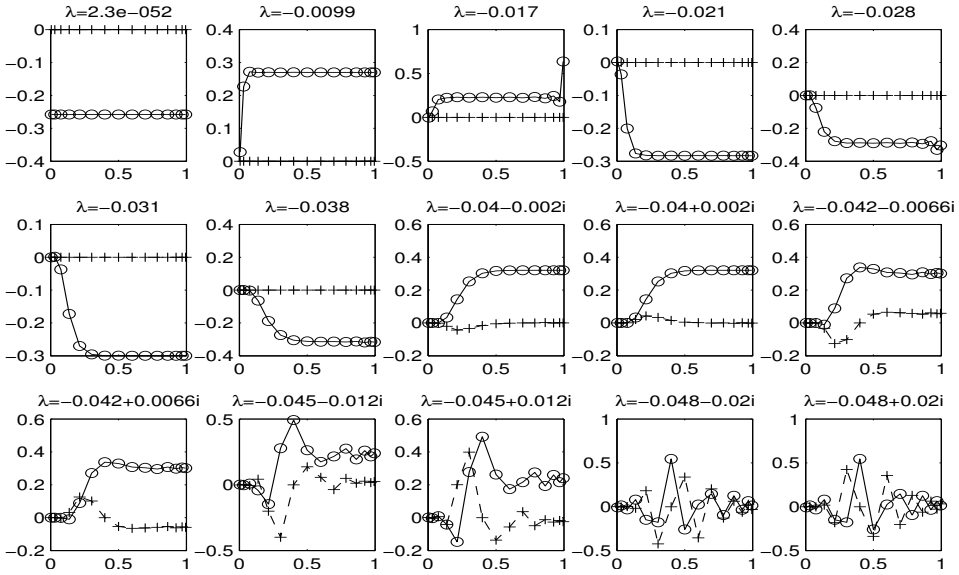


Figure 2.7: $S - \tilde{S}$: Real (o) and imaginary ($+$) components of each eigenvector at the collocation points, non-stiff case, $p = 15$, *SDC.*

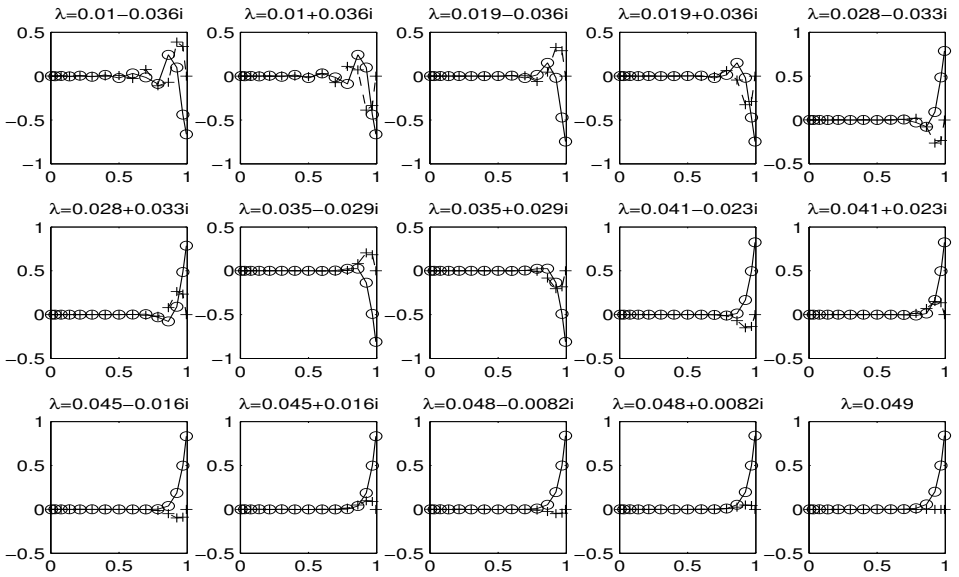


Figure 2.8: S: Real (o) and imaginary ($+$) components of each eigenvector at the collocation points, non-stiff case, $p = 15$, *Picard iteration*

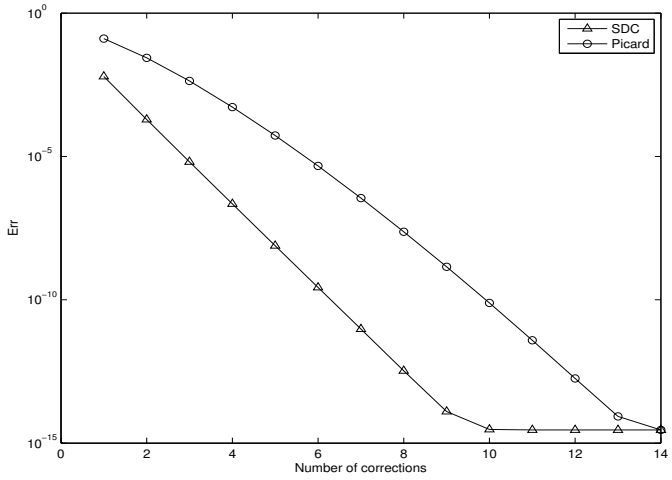


Figure 2.9: How errors decay after each SDC or Picard iteration.

$\tilde{\boldsymbol{\delta}}^{[0]} = \sum_{i=1}^{p} \alpha_i \mathbf{v}_i$, we can express k iterations of the correction matrix impacting the initial error as

$$\begin{aligned} C^k \tilde{\boldsymbol{\delta}}^{[0]} &= C^k(\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_p \mathbf{v}_p) \\ &= \Big(\alpha_1 (\lambda_1)^k \mathbf{v}_1 + \alpha_2 (\lambda_2)^k \mathbf{v}_2 + \cdots + \alpha_p (\lambda_p)^k \mathbf{v}_p\Big) \end{aligned} \tag{2.36}$$

where $C\mathbf{v}_i = \lambda_i \mathbf{v}_i$ with $|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_p|$. Though $|\lambda_p|$ for C_{ns}^p and C_{ns} are similar, the eigenvalues in Figs. 2.7 and 2.8 show different behaviors.

For the matrix $S - \tilde{S}$ in the SDC iterations, zero is an eigenvalue and the corresponding eigenvector is the constant vector. Notice that for both methods, when a Taylor expansion is applied to the error term in the initial provisional solution, the constant component is usually the largest term, followed by linear, then quadratic, and then higher degree terms. Thus, one should expect smaller initial error when using the SDC method because SDC can effectively eliminate the dominating "low-frequency" error components. This is validated numerically in Fig. 2.9, by implementing both the SDC and Picard iterations for the model problem $y'(t) = y(t) + f(t)$, where $f(t)$ is chosen so that the analytical solution is given by $y(t) = \frac{1}{1+t}$. The figure shows how the errors decay after each SDC or Picard iteration in one time step $[0, 0.6]$. In the simulation, $p = 15$ is used for both methods; and the spectral radius of $S - \tilde{S}$ is approximately 0.049. It can be seen that the error from the SDC iterations is smaller than that of the Picard iterations, and the asymptotic decay slope of the Picard iterations approaches that of the SDC method. Also, the numerical value

of the slope of the SDC curve is approximately -3.37, which is very close to the theoretical value $-3.53 \approx \log(0.6 \cdot 0.049)$.

When the SDC methods are applied to the **stiff** systems where $|\lambda \Delta t| \gg 1$, in Fig. 2.10, we plot all the eigenvectors of the correction matrix C_{s} for $p = 15$. It can be observed that higher

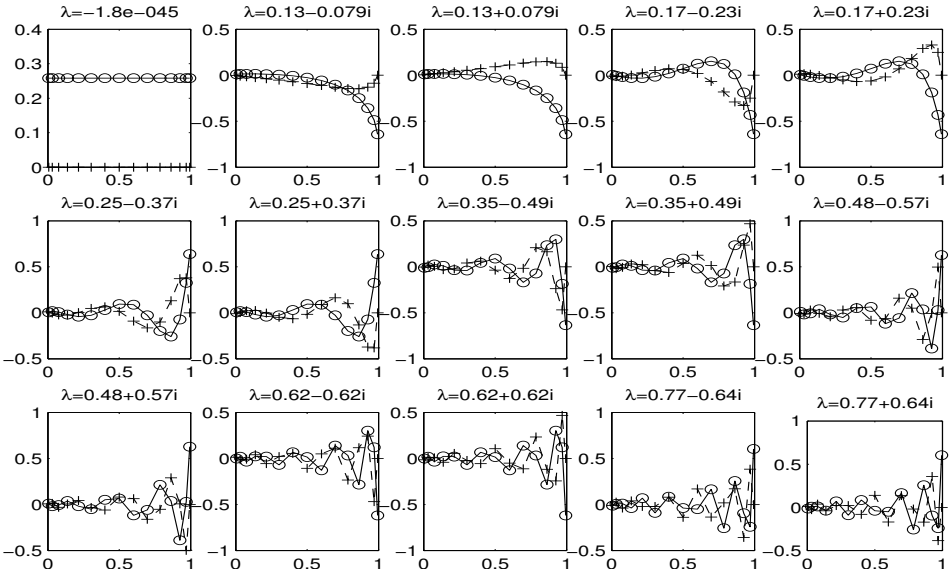


Figure 2.10: Real (o) and imaginary ($+$) components of each eigenvector at the collocation points, stiff case, $p = 15$, *backward Euler preconditioned Gauss collocation formulation.*

frequency errors decay slower than the lower frequency errors because the moduli of the corresponding eigenvalues are larger. Recall that for the initial provisional solution in the SDC iterations, the low frequency errors are usually the dominating components. The overall errors will therefore decay rapidly in the first few iterations, but "order reduction" or even "divergence" is expected eventually for a large number of corrections due to the asymptotic convergence properties determined by the spectral radius $\rho(C_{\mathrm{s}})$. One interesting numerical example can be found in Fig. 7 in [30], where the SDC method is applied to Andrews' squeezing DAE system. For this specific example and different step sizes, the errors decay in the first few iterations and start to increase once the dominating error becomes the high frequency component corresponding to the largest eigenvalue. In existing deferred correction implementations, such divergence (and order reduction for smaller p) was usually controlled by fixing the total number of iterations to bound the growth of the eigenvectors corresponding to eigenvalues of large moduli, and by using smaller step sizes to reduce the magnitude of the coefficients of these eigenvectors in the initial error.

Comment: For general DAE systems, it is usually expected that in the discretized algebraic equations, since $\tilde{S}^{-1}$ is applied to precondition S directly by applying the implicit function theorem, the convergence of the SDC method for DAE systems will most likely depend on the spectral radius of $I - \tilde{S}^{-1}S$, especially for higher index DAE systems, and the numerical properties of the SDC methods will be similar to the strongly stiff limit case for ODEs.

2.2.5 Different Deferred Correction Methods

In this subsection, we discuss several deferred correction strategies and present their properties. We focus on the **"yp-formulation"** but other formulations (like the integral formulation) have also been studied and can be included in the "deferred correction methods database". In the "convergence procedure", appropriate deferred correction schemes will be selected to reduce different error components in the initial solution for faster convergence to the collocation formulation.

We also studied the backward Euler preconditioned SDC type methods for the Radau IIa collocation formulation (SDC-Radau) where the right end point $t = \Delta t$ is included in the spectral integration, and the Lobatto formulation (SDC-Lobatto) with both end points $t = 0$ and $t = \Delta t$ used in the formulation.

For Radau IIa nodes, we found that the convergence behaviors of the SDC-Radau schemes are similar to those of the Gaussian nodes in both the non-stiff ($|\lambda\Delta t|$ small) and stiff ($|\lambda\Delta t|$ large) cases. In Table 2.2, we show the spectral radius $\rho(C)$ of the correction matrices for different numbers of Radau IIa nodes for the **stiff case**. It can be seen that when $p \geq 12$, the SDC-Radau methods become divergent. We also plot the convergence region of the SDC-Radau. Similar to the Gauss

p	2	3	4	5	6	7	8
$\rho(C)$	0.2500	0.4344	0.6184	0.7364	0.8161	0.8726	0.9146
p	9	10	11	12	13	14	15
$\rho(C)$	0.9469	0.9724	0.9931	1.0101	1.0244	1.0365	1.0470
p	16	17	18	19	20	25	50
$\rho(C)$	1.0560	1.0639	1.0709	1.0772	1.0827	1.1037	1.1444

Table 2.2: $\rho(C)$ for different numbers of nodes, *SDC-Radau.*

collocation case, our numerical results show that the methods are A-convergent for smaller p, but become divergent when p is large. Also, none of these formulations are L-convergent.

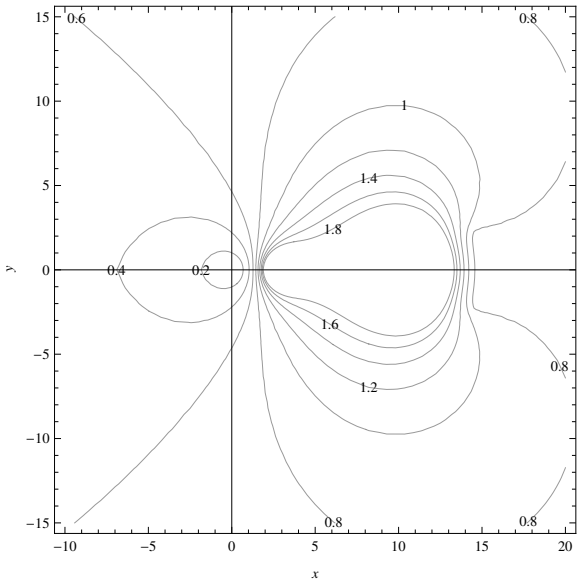


Figure 2.11: Contour of $\rho(C)$ for $p = 4$ for *SDC-Radau*, $\lambda\Delta t = x + iy$.

For Lobatto nodes, the left end point ($t = 0$) is included in the integration quadrature and we also add $t_0 = 0$ to the collocation formulation. It is easy to see that all entries in the first row of the integration matrix S (representing $\int_0^0 Y(\tau)d\tau$) will be zero. We denote

$$S = \begin{bmatrix} 0_{1\times 1} & \mathbf{0}_{1\times(p-1)} \\ S_{21} & S_{22} \end{bmatrix},$$

where S_{21} is the $(p-1)\times 1$ vector and S_{22} is the $(p-1)\times(p-1)$ submatrix. The equation at $t = 0$ is simply the initial consistency condition $\tilde{Y}_0 = f(t_0, y_0)$. The low order quadrature rule can be represented in a similar way as

$$\tilde{S} = \begin{bmatrix} 0_{1\times 1} & \mathbf{0}_{1\times(p-1)} \\ \tilde{S}_{21} & \tilde{S}_{22} \end{bmatrix}.$$

When the backward Euler's method (rectangular rule using the right end point) is used, $\tilde{S}_{21}$ is a zero vector, and $\tilde{S}_{22}$ contains the lengths of the subintervals between adjacent Lobatto quadrature nodes similar to Eq. (2.6). Applying the Woodbury matrix identity, the correction matrix can be

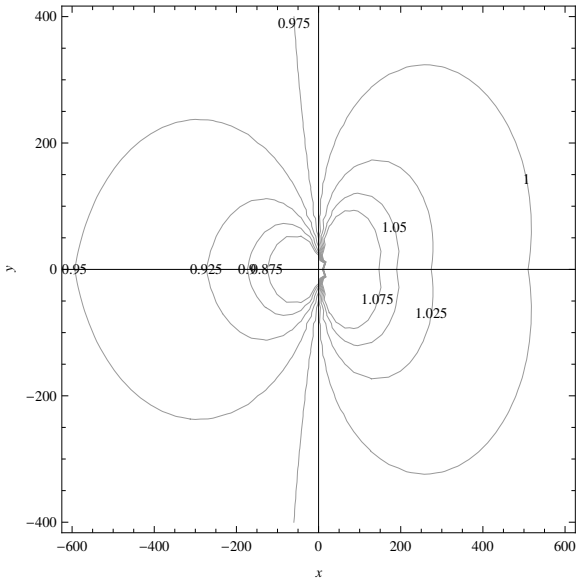


Figure 2.12: Contour of $\rho(C)$ for $p = 10$ for *SDC-Radau*, $\lambda\Delta t = x + iy$.

simplified as

$$
\begin{aligned}
C &= I - (I - \lambda\Delta t\tilde{S})^{-1}(I - \lambda\Delta tS) \\
&= I - \begin{bmatrix} 1 & \mathbf{0} \\ (I - \lambda\Delta t\tilde{S}_{22})^{-1}\lambda\Delta t\tilde{S}_{21} & (I - \lambda\Delta t\tilde{S}_{22})^{-1} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ -\lambda\Delta tS_{21} & (I - \lambda\Delta tS_{22}) \end{bmatrix} \\
&= \begin{bmatrix} 0 & \mathbf{0} \\ (I - \lambda\Delta t\tilde{S}_{22})^{-1}\lambda\Delta t(S_{21} - \tilde{S}_{21}) & I - (I - \lambda\Delta t\tilde{S}_{22})^{-1}(I - \lambda\Delta tS_{22}) \end{bmatrix}.
\end{aligned}
$$

One therefore only needs to study the "sub-correction matrix" $I - (I - \lambda\Delta t\tilde{S}_{22})^{-1}(I - \lambda\Delta tS_{22})$ to understand the convergence properties of the original correction matrix. For stiff systems when $|\lambda\Delta t|$ is large, one needs to study the matrix $I - \tilde{S}_{22}^{-1}S_{22}$. In Table 2.3, we show the spectral radius of this matrix for stiff ODE systems. Similar to the Gaussian and Radau IIa cases, the SDC-Lobatto methods become divergent when $p > 14$ and order reduction is expected for smaller numbers of nodes. For comparison, we also plot the convergence regions of SDC-Lobatto methods.

Comment: In most existing analysis and implementations of deferred correction methods, a fixed number of iterations is performed and the resulting "solution" may still be far away from the converged solution in each time step. Hence, one should expect a relatively large error in the initial value y_0 for the next step. For stiff problems, the large error may accumulate rapidly when

p	3	4	5	6	7	8	9
$\rho(C)$	0.5000	0.5922	0.6837	0.7576	0.8150	0.8600	0.8957
p	10	11	12	13	14	15	16
$\rho(C)$	0.9247	0.9485	0.9685	0.9853	0.9998	1.0123	1.0233
p	17	18	19	20	21	25	50
$\rho(C)$	1.0330	1.0415	1.0492	1.0560	1.0622	1.0820	1.1333

Table 2.3: $\rho(C)$ for different numbers of nodes, *SDC-Lobatto methods.*

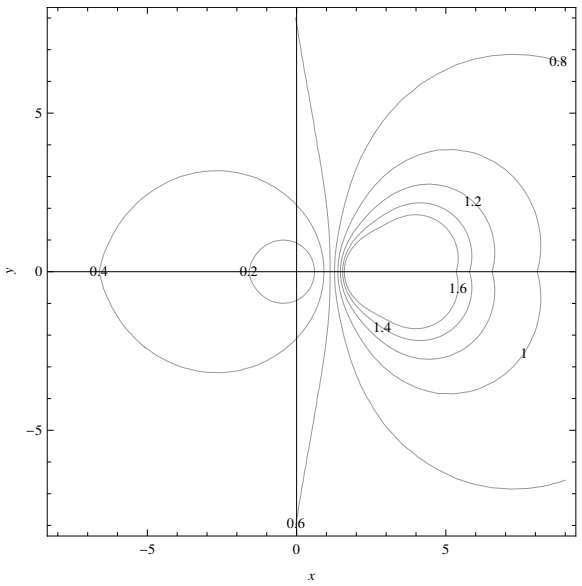


Figure 2.13: Contour of $\rho(C)$ for $p = 4$, SDC-Lobatto methods, $\lambda\Delta t = x + iy$.

the number of time steps increases in any yp-formulation using the left end point $t = 0$. This can be shown by studying the initial provisional solution $\mathbf{Y}^{[0]} = (I - \lambda\Delta t\tilde{S})^{-1}(\lambda\mathbf{y}_0 + \mathbf{F})$ (also see Eq. (2.20)). When the left end point is used in the collocation formulation, as

$$(I - \lambda\Delta t\tilde{S})^{-1} = \begin{bmatrix} 1 & \mathbf{0} \\ (I - \lambda\Delta t\tilde{S}_{22})^{-1}\lambda\Delta t\tilde{S}_{21} & (I - \lambda\Delta t\tilde{S}_{22})^{-1} \end{bmatrix},$$

the error in the first entry (corresponding to the left end point) of $\mathbf{Y}^{[0]}$ will be λ times the error from the initial value y_0. When this entry is used in the spectral integration scheme, this error will propagate to other collocation points and magnify the overall error by $O(\lambda)$ in the final solution at each time step, resulting in an unstable numerical time marching scheme. Therefore, the yp-formulation with the left end point $t = 0$ should be avoided in the standard deferred correction

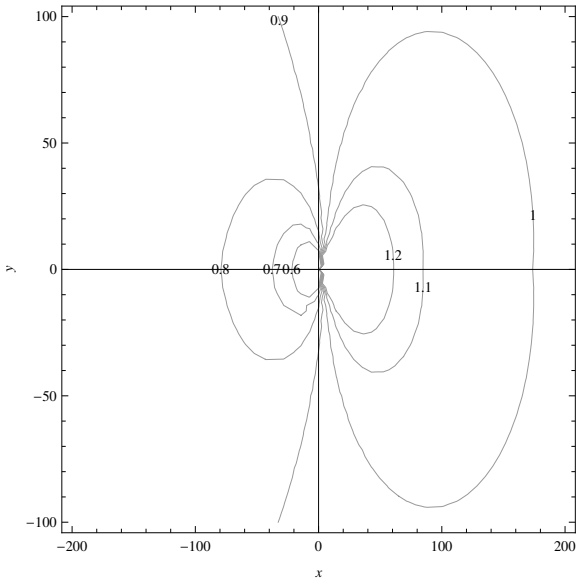


Figure 2.14: Contour of $\rho(C)$ for $p = 10$, SDC-Lobatto methods, $\lambda\Delta t = x + iy$.

methods.

It is well-known that the uniform interpolations suffer from the Runge phenomena when a large number of interpolation points are used, so in existing implementations, only low order uniform collocation formulations (e.g., $p < 10$) are considered in the integral deferred correction (InDC) methods [14]. In this subsection, we analyze the backward Euler preconditioned deferred correction methods for the uniform yp-collocation formulations (denoted as InDC-yp). In Figures 2.15-2.17, we show the convergence regions for $p = 4$, $p = 5$ and $p = 10$. The numerically computed convergence regions show that when $p = 4$, the method is A-convergent. However, the method is no longer A-convergent when $p > 4$.

The most interesting feature of the InDC-yp is the following theorem for **stiff** systems.

Theorem 2.2.3. *For the InDC-yp method, when $|\lambda\Delta t| \to \infty$, the correction matrix $\tilde{S}^{-1}S - I$ has eigenvalues equal to zero; and its Jordan canonical form consists of one Jordan block.*

The proof is sketched in the Appendix. Because there only exist zero eigenvalues, we conclude that the InDC-yp methods are L-convergent for $p < 5$. Clearly, the InDC methods have better convergence properties, but larger error is expected from the converged solution due to the uniform collocation points for large p.

In the following we study the convergence properties of the second order trapezoidal rule

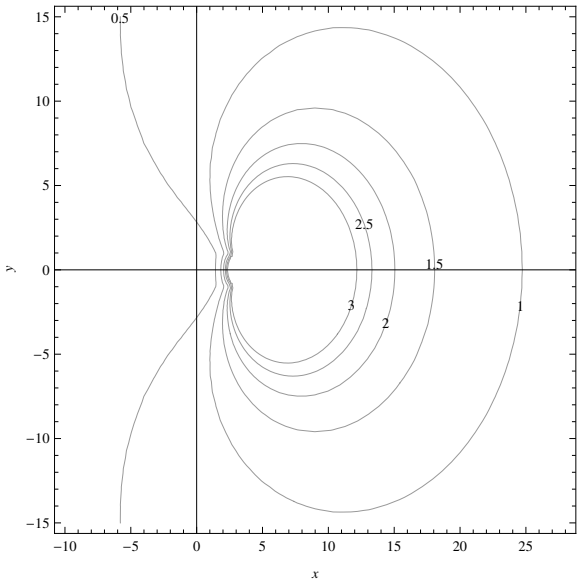


Figure 2.15: Contour of $\rho(C)$ for $p = 4$ for *InDC-yp*, $\lambda\Delta t = x + iy$.

preconditioned yp-formulations.

We first consider the **non-stiff** case. The left end point ($t = 0$) is used in the first subinterval by the trapezoidal rule. We add it to the collocation formulation to compare the trapezoidal rule preconditioned Lobatto collocation formulation (denoted as SDC-Lobatto-T and the corresponding correction matrix is denoted as C_{ns}^T) with the backward Euler preconditioned Lobatto collocation formulation, SDC-Lobatto. From Fig. 2.18, it can be seen that the spectral radius of $S - \tilde{S}$ from C_{ns}^T is smaller than that from the SDC-Lobatto. Therefore for non-stiff problems, the second order trapezoidal rule preconditioned SDC-Lobatto-T should converge asymptotically faster. Also, using the trapezoidal rule predictor, the initial low order solution should have much better accuracy (smaller error). One interesting observation is that as the spectral radius of the trapezoidal rule preconditioned SDC-Lobatto-T method is non-zero, one should only expect the error to decay by the factor $\lambda\Delta t$ after each iteration, assuming the initial error has all eigenmodes. This disagrees with some existing claims that the error decays by a factor Δt^2 after each 2^{nd} order SDC correction. Such disagreements were also pointed out in [14], where the integral deferred correction methods are studied as special Runge-Kutta approaches.

For **stiff** problems, the second order trapezoidal rule preconditioned SDC-Lobatto-T iterations show worse convergence properties. In Table 2.4, we show the spectral radius of the correction

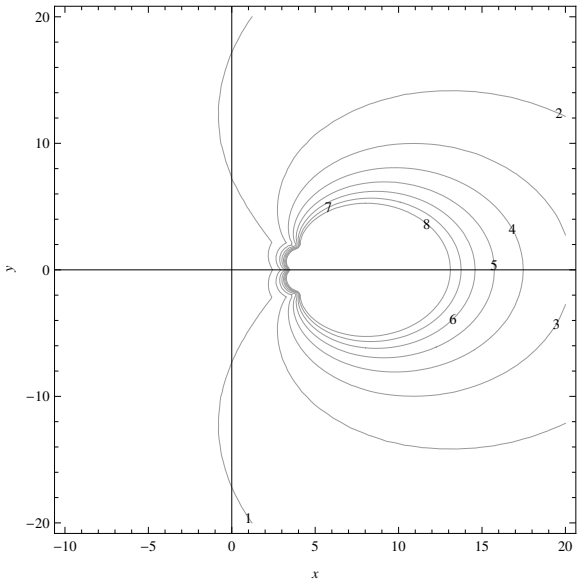


Figure 2.16: Contour of $\rho(C)$ for $p = 5$ for *InDC-yp*, $\lambda\Delta t = x + iy$.

matrix in this regime. It can be seen that the trapezoidal rule preconditioned SDC iterations become divergent when $p > 5$. Therefore, without resolving the "order reduction" and divergence problems, the higher order trapezoidal rule preconditioner is usually not recommended for solving the pseudo-spectral discretization for stiff ODE and DAE systems.

p	3	4	5	6	7	8	9
$\lvert\lambda\rvert_{max}$	0.3333	0.6180	0.8934	1.1658	1.4370	1.7076	1.9780
p	10	11	12	13	14	15	16
$\lvert\lambda\rvert_{max}$	2.2482	2.5183	2.7884	3.0585	3.3285	3.5986	3.8687
p	17	18	19	20	21	25	50
$\lvert\lambda\rvert_{max}$	4.1388	4.4089	4.6789	4.9490	5.2191	6.2995	13.0530

Table 2.4: $\rho(C)$ of SDC-Lobatto-T, strongly stiff limit case.

Another interesting observation is obtained when the trapezoidal rule preconditioner is applied to the uniform collocation formulation (denoted as InDC-yp-T) of non-stiff problems, described in the following theorem, and the proof is given in the Appendix.

Theorem 2.2.4. *For a non-stiff ODE system and its uniform collocation discretization, after each trapezoidal rule preconditioned InDC-yp-T iteration, the error decays by the factor* $(\Delta t)^2$ *before reaching its discretization order* $(\Delta t)^{p+1}$.

Therefore, higher order preconditioners are more effective to reduce the non-stiff errors when the

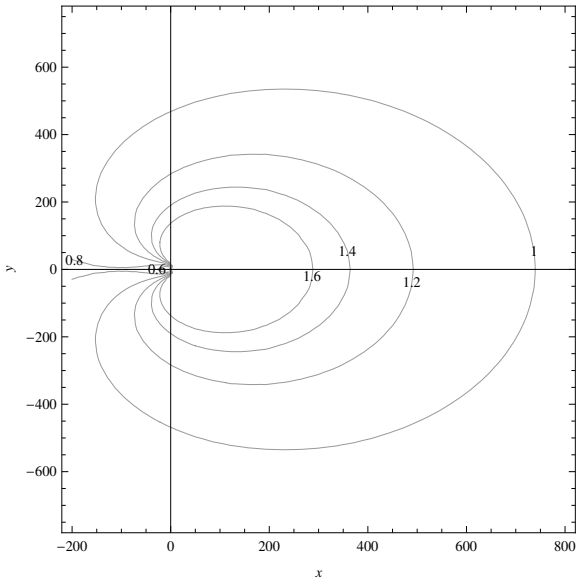


Figure 2.17: Contour of $\rho(C)$ for $p = 10$ for *InDC-yp*, $\lambda\Delta t = x + iy$.

uniform nodes are used. However many of these schemes show worse convergence properties for stiff systems in the standard deferred correction iterations, e.g., we found that for $p = 6$, the trapezoidal rule preconditioned iterations are divergent. For smaller p, severe order reduction is observed.

For non-stiff problems, existing numerical results show that the Neumann-series type deferred correction methods are very effective in the solution procedure to converge to the corresponding collocation formulation. This is unfortunately not true for stiff problems, and one has to deal with the divergence and order reduction for stiff ODE systems in the convergence procedure. One effective solution in existing literature is to search for the optimal solution in the Krylov subspace. One can use the Krylov deferred correction (KDC) methods [29, 30] to solve the preconditioned formulation in Eq. (2.30). For **linear stiff** problems, instead of the Neumann series solution in Eq. (2.28), one can search for the optimal least squares solution in the Krylov subspace $\mathcal{K}_m(C, \mathbf{Y}^{[0]}) = span\{\mathbf{Y}^{[0]}, C\mathbf{Y}^{[0]}, C^2\mathbf{Y}^{[0]}, \cdots, C^{m-1}\mathbf{Y}^{[0]}\}$ using existing Krylov subspace methods such as the GMRES (generalized minimal residuals) or BiCGStab (bicongugate gradient stabilized method) as the matrix C is usually non-symmetric [4, 35, 46].

For **nonlinear stiff** problems, one can apply the Jacobian-free Newton Krylov (JFNK) methods to find the root of the low-order method preconditioned system $\tilde{\delta} = \mathbf{H}(\tilde{\mathbf{Y}})$, where the "input" variable $\tilde{\mathbf{Y}}$ is the approximate solution and the "output" $\tilde{\delta}$ is the low-order estimate of the error in

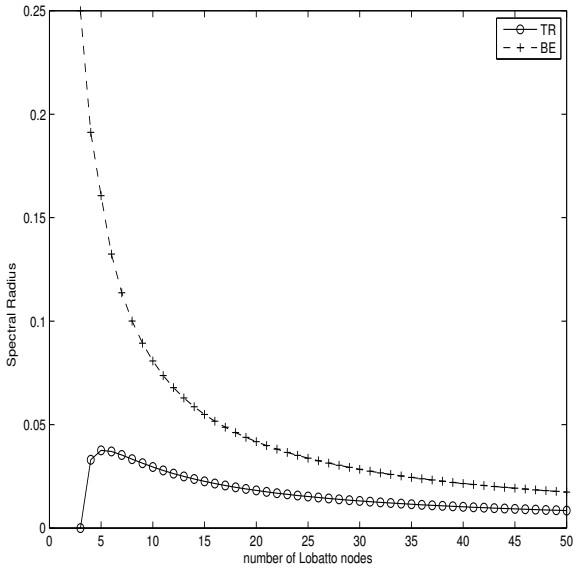


Figure 2.18: Spectral Radius $\rho(S - \tilde{S})$ for different numbers of nodes, *SDC-Lobatto and SDC-Lobatto-T.*

the SDC correction. Note that when $\tilde{\mathbf{Y}}$ solves the original collocation formulation in Eq. (2.2), the output $\tilde{\delta} = \mathbf{0}$. Also, when the output is a good estimate of the error in the input variable $\tilde{\mathbf{Y}}$, by applying the implicit function theorem, one can show that the Jacobian matrix of $\mathbf{H}$ is close to $-I$. We refer interested readers to [36, 37] for details of the JFNK methods. In the following we present the algorithmic structure of one step of the KDC methods marching from 0 to Δt using existing implementations of the JFNK methods.

Krylov deferred correction method: Subroutine OneStep($y(t_0 + \Delta t)$, $\mathbf{Y}$, $y(t_0)$, t_0, Δt)

Comment:

Input: Initial values $y(t_0)$ at $t = t_0$ and step size Δt.

Output: Solution $y(t_0 + \Delta t)$ at $t_0 + \Delta t$ and derivatives $\mathbf{Y}$ at collocation nodes.

Step 1, Predictor: Use a low order method to find an approximate solution $\tilde{\mathbf{Y}}$ as the initial guess.

Step 2, JFNK: Call existing JFNK solver to find the root $\mathbf{Y}$ of the equation $\tilde{\delta} = \mathbf{H}(\tilde{\mathbf{Y}}) = \mathbf{0}$.

Step 3, Output: Use high order quadrature and integrate $\mathbf{Y}$ to get $y(t_0 + \Delta t)$.

In the JFNK method, the function evaluation $\tilde{\delta} = \mathbf{H}(\tilde{\mathbf{Y}})$ is simply one SDC iteration for the

given provisional solution $\tilde{\mathbf{Y}}$, and such a function evaluation module should be provided by the user. We refer interested readers to [29, 30, 33] for details of the KDC algorithm and preliminary numerical results. Though KDC is a promising method, we do find that straightforward application of existing JFNK packages in KDC is not optimal. For small Δt, existing JFNK methods often encounter difficulty converging to the collocation formulation even though the original deferred correction approaches converge satisfactorily. Also, for some settings, the deferred correction approach converges faster than the JFNK. We believe the reason is that the general purpose JFNK solvers are unaware of the special structures in the preconditioned system implicitly given by the function $\mathbf{H}$. Modification and optimization of the JFNK methods for the numerical framework will be further addressed in chapter 3.

2.2.6 Integral Formulation, yp-formulation, and Convergence

Our analysis also shows that using different formulations will also change the convergence properties of the deferred correction iterations. In this subsection, we compare the integral formulation with the yp-formulation for the linear ODE $y'(t) = \lambda y(t) + f(t)$ for both non-stiff and stiff cases. In the **yp-formulation**, we use $Y(t) = y'(t)$ as the unknown and solve the discretized system in Eq. (2.16). The iterations for the yp-formulation is given in Eq. (2.28), and the converged solution $\mathbf{Y}$ is explicitly given by

$$\mathbf{Y} = (I - \lambda \Delta t S)^{-1}(\lambda \mathbf{y}_0 + \mathbf{F}).$$

After finding $\mathbf{Y}$, the solution $\tilde{\mathbf{y}}$ is constructed using $\tilde{\mathbf{y}} = \mathbf{y}_0 + \Delta t S \mathbf{Y}$. In the **integral formulation**, one computes $y(t)$ directly by solving the Picard integral equation $y(t) = y_0 + \int_0^t (\lambda y(\tau) + f(\tau))\, d\tau$. The discretized system is given by $\tilde{\mathbf{y}} = \mathbf{y}_0 + \Delta t S\,(\lambda \tilde{\mathbf{y}} + \mathbf{F})$, and the converged solution is given explicitly by

$$\tilde{\mathbf{y}} = (I - \lambda \Delta t S)^{-1}(\mathbf{y}_0 + \Delta t S \mathbf{F}).$$

The Neumann series expansion for the preconditioned formulation

$$(I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t S)\tilde{\mathbf{y}} = (I - \lambda \Delta t \tilde{S})^{-1}(\mathbf{y}_0 + \Delta t S \mathbf{F}) = \mathbf{y}^{[0]}$$

is given by

$$\mathbf{y}^{[n]} = \mathbf{y}^{[0]} + C\mathbf{y}^{[0]} + C^2\mathbf{y}^{[0]} + \cdots + C^n\mathbf{y}^{[0]}$$

where C is the same correction matrix as in the yp-formulation.

However, after a fixed number K iterations, the truncated expansions will have different properties. Assuming both series expansions are convergent, the error from the truncated yp-formulation is then given by

$$err_{yp} = \Delta t S(\sum_{k=K+1}^{\infty} C^k \tilde{\mathbf{Y}}^{[0]}) = \Delta t S \left(\sum_{k=K+1}^{\infty} C^k (I - \lambda \Delta t \tilde{S})^{-1} (\lambda \mathbf{y}_0 + \mathbf{F}) \right),$$

and the error from the integral formulation is given by

$$err_{integral} = \sum_{k=K+1}^{\infty} C^k \tilde{\mathbf{y}}^{[0]} = \sum_{k=K+1}^{\infty} C^k \left((I - \lambda \Delta t \tilde{S})^{-1} (\mathbf{y}_0 + \Delta t S \mathbf{F}) \right).$$

Comparing the error terms, we can see that for non-stiff problems when $|\lambda \Delta t| \ll 1$, the error from the yp-formulation should be one order higher (in Δt) than the integral formulation due to the additional Δt factor. However for stiff problems when $|\lambda \Delta t| \gg 1$, the integral form is preferred. Also, when the deferred correction methods are applied to the integral formulations with the left end point $t = 0$, the numerical schemes should be more stable in time marching than the corresponding yp-formulation case discussed in Sec. 2.2.5, since the term $\lambda \mathbf{y}_0$ doesn't exist in the integral formulation.

2.3 Algorithm Design Guidelines and Numerical Experiments

In most existing deferred correction implementations, one applies a particular deferred correction method for the corresponding collocation formulation. For stiff systems, when the estimated error is still large after a fixed number of iterations due to the order reduction or divergence, a commonly used strategy is to reduce the step size since the error components corresponding to the "bad eigenvalue" in the provisional solution become smaller when Δt decreases. One can therefore "control" the growth of the divergent or slowly convergent components in the Neumann series expansion by stopping the iterations before they become significant. The drawback of this strategy

is that this approach only works when the step size is reasonably small (due to the divergence or order reduction), and one can no longer take advantage of the large step size in the optimal collocation formulations.

In the new numerical framework, instead of using one single deferred correction method for a particular collocation formulation, different deferred correction techniques can be applied to reduce different components in the error of the provisional solution in order to more efficiently converge to the solution of the "optimal" collocation formulation for the underlying ODE system. In the following, we provide some guidelines for each step of the numerical framework. Preliminary numerical experiments are also performed to support these guidelines. We want to mention that the new perspective of looking at the deferred correction methods as iterative schemes to converge to the optimal collocation formulation also allows the introduction of other existing effective preconditioning techniques for faster convergence, e.g., domain decomposition or multigrid techniques commonly used in today's spatial solvers.

2.3.1 Optimal Collocation Formulation

p	4	5	6	7	8	9	10
Err_U	1.51e-1	1.05e+0	4.82e-2	5.27e-1	1.78e-2	2.68e-1	7.11e-3
Err_G	3.83e-2	2.23e-2	5.91e-3	4.11e-3	1.05e-3	7.99e-4	1.99e-4
p	11	12	13	14	15	16	17
Err_U	1.36e-1	2.97e-3	6.94e-2	1.28e-3	3.52e-2	5.61e-4	1.78e-2
Err_G	1.57e-4	3.86e-5	3.11e-5	7.55e-6	6.17e-6	1.48e-6	1.23e-6
p	18	19	20	21	25	31	41
Err_U	2.50e-4	9.03e-3	1.13e-4	4.56e-3	1.15e-3	1.47e-4	4.66e-6
Err_G	2.93e-7	2.44e-7	5.81e-8	4.86e-8	1.94e-9	1.54e-11	4.88e-15

Table 2.5: Errors from Gauss and uniform collocation formulations for different numbers of nodes.

A good collocation formulation can be selected from the "collocation formulation database" based on the physical properties of the system. For ODE systems, our default choice is the Legendre polynomial based Gauss collocation formulation. In general, the orthogonal basis functions based collocation formulations are recommended because it is a widely accepted fact that they outperform the uniform nodes based formulations by allowing larger step sizes and better accuracy. This is demonstrated by comparing the solutions from the Gauss and uniform collocation formulations for

the non-stiff ODE system $y'(t) = y(t) + f(t)$ with the analytical solution $y(t) = \frac{1}{1+5(x-0.5)^2}$ (and $f(t)$ is determined accordingly). In Table 2.5, we list the errors for different numbers of nodes for both formulations, where the numerical solution is derived by solving the collocation formulations directly using Gauss elimination (instead of deferred correction iterations) in one time step $[0, 1]$. It can be seen that for this particular system, the results from the Gauss collocation formulations are always more accurate than those from the uniform collocation formulations.

There are several factors in finding the optimal formulation for a specific ODE system. One may need to know the properties of the solution to determine which formulation will need fewer points for the same accuracy requirement. In general, the orthogonal basis based collocation formulations or the skeletonization based schemes should give good results for most problems and uniform collocation formulations should be avoided, especially when one wants to use a large number of node points for efficiency considerations.

2.3.2 Techniques for Convergence Procedure

Existing studies of the deferred correction methods show that it is more efficient to solve the collocation formulation using an iterative approach instead of the direct Gauss elimination, and the low-order methods are good preconditioners for the pseudo-spectral collocation formulation. In the "convergence procedure" of the numerical framework, different preconditioning techniques can be integrated to eliminate the errors of the provisional solutions efficiently. In this section, we compare different strategies for stiff and non-stiff problems, and provide guidelines for faster convergence.

We first compare different schemes for the non-stiff model problem $y'(t) = y(t) + f(t)$ with analytical solution $y(t) = \frac{1}{1+t}$ (and $f(t)$ determined accordingly). In Table 2.6, we show how the errors change for different numbers of deferred correction iterations using different low-order preconditioners and collocation schemes. We march from $t = 0$ to $t_{final} = 3$ using "nsteps" time steps, and set the number of node points to $p = 7$ for each time step in all cases. In the "Uniform+B(oth)" collocation formulation, both end points are used in the formulation. We also tested the Radau IIa nodes and Gaussian nodes, and the results are very similar to those from the Lobatto collocation formulation for the non-stiff case in Table 2.6. We therefore neglect those results in the table. It can be seen that:

nsteps	4	8	16	32	Order
yp, BE, Uniform+B, 0 SDC Iters	3.14e-1	7.55e-2	1.83e-2	4.48e-3	2.04
yp, BE, Uniform+B, 1 SDC Iters	2.64e-2	2.72e-3	3.06e-4	3.62e-5	3.17
yp, BE, Uniform+B, 2 SDC Iters	2.31e-3	1.00e-4	5.19e-6	2.94e-7	4.31
yp, BE, Lobatto, 0 SDC Iters	3.78e-1	9.39e-2	2.32e-2	5.76e-3	2.01
yp, BE, Lobatto, 1 SDC Iters	4.56e-2	4.93e-3	5.71e-4	6.85e-5	3.13
yp, BE, Lobatto, 2 SDC Iters	6.43e-3	2.80e-4	1.48e-6	8.53e-7	4.29
yp, TR, Uniform+B, 0 SDC Iters	1.49e-2	1.88e-3	2.28e-4	2.78e-5	3.02
yp, TR, Uniform+B, 1 SDC Iters	6.90e-5	1.62e-6	4.36e-8	1.30e-9	5.23
yp, TR, Uniform+B, 2 SDC Iters	3.11e-5	2.56e-7	1.33e-9	6.43e-12	7.42
yp, TR, Lobatto, 0 SDC Iters	2.18e-2	3.11e-3	4.01e-4	5.03e-5	2.92
yp, TR, Lobatto, 1 SDC Iters	7.44e-5	5.25e-6	3.74e-7	2.50e-8	3.84
yp, TR, Lobatto, 2 SDC Iters	2.74e-6	7.31e-8	2.25e-9	6.92e-11	5.08
integral, BE, Uniform+B, 0 SDC Iters	6.20e-1	2.78e-1	1.32e-1	6.45e-2	1.08
integral, BE, Uniform+B, 1 SDC Iters	5.38e-2	1.01e-2	2.22e-3	5.22e-4	2.23
integral, BE, Uniform+B, 2 SDC Iters	5.44e-3	4.00e-4	3.89e-5	4.32e-6	3.43
integral, BE, Lobatto, 0 SDC Iters	8.38e-1	3.71e-1	1.75e-1	8.46e-2	1.10
integral, BE, Lobatto, 1 SDC Iters	9.88e-2	1.89e-2	4.15e-3	9.74e-4	2.22
integral, BE, Lobatto, 2 SDC Iters	1.52e-2	1.11e-3	1.08e-4	1.19e-5	3.43
integral, TR, Uniform+B, 0 SDC Iters	1.55e-2	3.89e-3	9.73e-4	2.43e-4	2.00
integral, TR, Uniform+B, 1 SDC Iters	1.47e-5	1.17e-6	9.78e-8	6.46e-9	3.70
integral, TR, Uniform+B, 2 SDC Iters	3.08e-5	2.52e-7	1.29e-9	5.80e-12	7.46
integral, TR, Lobatto, 0 SDC Iters	2.62e-2	7.04e-3	1.80e-3	4.52e-4	1.96
integral, TR, Lobatto, 1 SDC Iters	6.08e-5	2.76e-6	1.39e-7	7.76e-9	4.31
integral, TR, Lobatto, 2 SDC Iters	7.98e-7	1.42e-8	9.46e-9	5.98e-10	3.50

Table 2.6: Errors and Orders of the backward Euler and trapezoidal rule preconditioned deferred correction iterations for different collocation formulations, non-stiff case.

(a) The order of the yp-formulation is 1 order higher than the corresponding integral formulation.

(b) After each correction, the backward Euler preconditioned deferred correction methods improve the convergence order by 1 for both the yp-formulation and integral formulation.

(c) For both the yp-formulation and integral formulation, the trapezoidal rule preconditioned deferred correction methods improve the convergence order by 2 after each iteration for the uniform collocation formulations. This is not true for the Lobatto nodes.

(d) For all cases, the trapezoidal rule preconditioner outperforms the backward Euler preconditioner for the non-stiff problem.

These results agree with our analysis in previous sections and suggest the following strategies to start the iteration procedure: (1) one should apply a high order "predictor" to uniform collocation formulations to derive a more accurate initial provisional solution $\mathbf{Y}^{[0]}$ using the yp-formulation; (2) to reduce the non-stiff error components in the provisional solution, the higher order method (e.g., trapezoidal rule) preconditioned deferred correction schemes for the yp-formulation with uniform grids are preferred as they show better convergence properties; (3) one should compare the result $\tilde{\delta}^{[0]}$ from the first deferred correction iteration to the initial provisional solution, to check if $\mathbf{Y}^{[0]}$ is an acceptable initial guess for the Newton's method to converge to the collocation formulation solution. One possible measure is to check if the ratio $||\tilde{\delta}^{[0]}||/||\mathbf{Y}^{[0]}||$ is sufficiently small; and (4) for the first several deferred correction iterations, as the dominating error comes from the non-stiff part, it is probably unnecessary to search for the solution in the Krylov subspace, and the fixed point type iterations (Neumann series for linear problems) should provide good convergence properties. This can be measured by the ratio of $||\tilde{\delta}^{[n+1]}||/||\tilde{\delta}^{[n]}||$. When the ratio is small, standard deferred correction iterations should still be acceptable.

A relatively large ratio $||\tilde{\delta}^{[n+1]}||/||\tilde{\delta}^{[n]}||$ (e.g. $> 1/2$) suggests that the dominating error no longer comes from the non-stiff components, and algorithms which can efficiently reduce the errors from the stiff components should be applied. It is unfortunately still an open problem what the **optimal** strategy should be to reduce the errors from the stiff components. In this dissertation, we consider possible strategies for two scenarios: (1) when only the Neumann series type iterations are used as in standard deferred correction procedures, and (2) when the Krylov subspace based iterative

methods can be applied to further accelerate the convergence. Note that many researchers prefer the standard deferred correction methods in the first scenario since it doesn't require additional overhead operations (e.g., solving the least squares problem using the Krylov subspace methods) or additional memory to store the vectors in the Krylov subspace. However when scenario (1) is used to solve stiff ODE systems, serious order reduction (or even divergence) is expected unless very small time step sizes are used. In the remainder of this chapter, we provide some guidelines for scenario (1); and we will present the framework for scenario (2) in chapter 3.

In Table 2.7, we check the numerical properties of different deferred correction schemes for the stiff model problem $y'(t) = \lambda y(t) + f(t)$ with analytical solution $y(t) = \frac{1}{1+t}$ (and $f(t)$ determined accordingly). We set $\lambda = -10^5$ and use the same settings for other parameters as in the non-stiff case. We show how the errors change for different numbers of deferred corrections in a time marching scheme. In the table, we add the "uniform+R" collocation formulation where only the right hand side is included in the spectral integration. We focus on the first order backward Euler preconditioner, and neglect results from the trapezoidal rule based schemes due to their poor convergence properties in the "strongly stiff limit" case as summarized in Table 2.4. The purpose of this experiment is not to identify which method should be used to reduce the stiff components errors, but to find out which methods should be avoided when standard deferred correction methods are preferred. We consider the case when one doesn't require the iteration procedure to converge to the collocation formulation and hence allows the existence of relatively large errors in the solution. Our observations can be summarized as follows:

(a) Without converging to the collocation formulation, the deferred correction schemes for the yp-formulation using the left end point should be avoided because the large error in the initial value will be magnified by the factor λ and will propagate to later steps when marching in time, as discussed in Sec. 2.2.5.

(b) When the iterations converge to an acceptable accuracy, the yp-formulation without the left end point will become acceptable (see the case *yp, Radau IIa, 2 SDC Iters*).

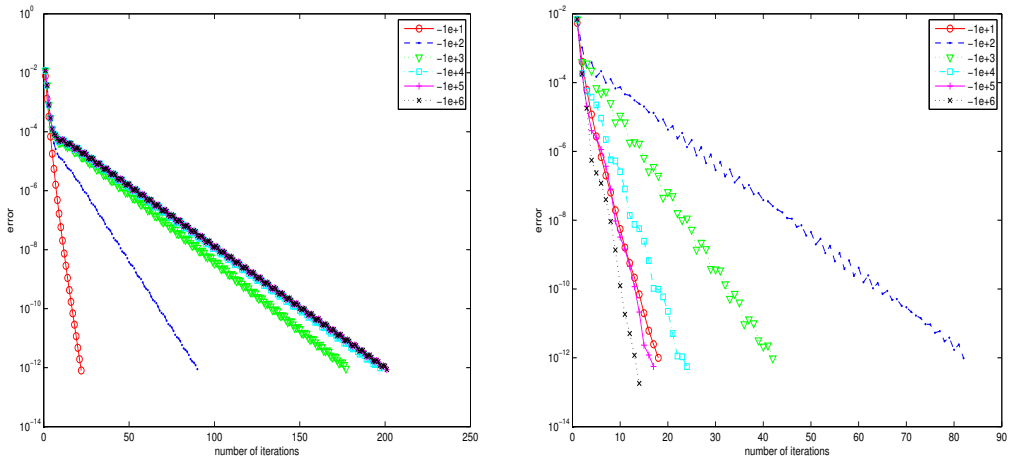
(c) When there are large errors in the initial solution, the integral formulations give more stable results than the yp-formulation, as discussed in Sec. 2.2.5.

nsteps	4	8	16	32	Order
yp, Uniform+B, 0 SDC Iters	2.04e+9	9.89e+20	1.17e+42	3.80e+79	—-
yp, Uniform+B, 2 SDC Iters	7.09e+6	2.86e+17	3.12e+36	1.11e+70	—-
yp, Lobatto, 0 SDC Iters	1.62e-1	2.84e-1	2.31e+0	4.98e+2	—-
yp, Lobatto, 2 SDC Iters	1.80e+5	9.16e+12	1.26e+26	5.81e+47	—-
yp, Uniform+R, 0 SDC Iters	6.21e-1	8.61e+0	4.69e+3	4.46e+9	—-
yp, Uniform+R, 2 SDC Iters	2.73e-2	1.39e-1	3.24e+1	1.64e+7	—-
yp, Radau IIa, 0 SDC Iters	1.42e-1	2.57e-1	2.21e+0	5.31e+2	—-
yp, Radau IIa, 2 SDC Iters	1.25e-4	2.61e-5	6.01e-6	1.44e-6	2.14
integral, Uniform+B, 0 SDC Iters	3.99e-3	1.97e-3	9.82e-4	4.90e-4	1.01
integral, Uniform+B, 2 SDC Iters	3.87e-4	1.50e-4	6.71e-5	3.17e-5	1.20
integral, Lobatto, 0 SDC Iters	7.38e-4	1.41e-4	2.77e-5	3.78e-6	2.51
integral, Lobatto, 2 SDC Iters	1.14e-3	6.12e-5	3.00e-5	2.34e-6	2.78
integral, Uniform+R, 0 SDC Iters	3.41e-3	1.69e-3	8.41e-4	4.19e-4	1.01
integral, Uniform+R, 2 SDC Iters	4.01e-6	4.66e-7	4.93e-8	9.87e-10	3.92
integral, Radau IIa, 0 SDC Iters	7.96e-4	3.97e-4	1.99e-4	9.99e-5	1.00
integral, Radau IIa, 2 SDC Iters	1.90e-4	7.49e-5	3.26e-5	1.50e-5	1.22

Table 2.7: Errors and orders of the backward Euler preconditioned deferred correction iterations for different collocation formulations, stiff case.

(d) The best results are derived using the integral formulation with uniform grids.

Note that in the previous numerical experiments, we follow the standard deferred correction schemes and consider both the converged and non-converged solutions in the simulations. Also, in the initial error, we have both stiff and non-stiff components. Because in the new numerical framework we first reduce the errors from the non-stiff components, it is more appropriate to focus on the rate of convergence (determined by the spectral radius of the correction matrix) for different schemes for stiff problems (instead of checking the errors in the first few iterations that also include the initial errors as in the previous experiments). In Figure 2.19, we compare the rate of convergence for the backward Euler preconditioned deferred correction iterations for the integral formulations using the **Gauss collocation points** to that using the **uniform collocation points**. Both schemes are applied to the model problem $y'(t) + \sin(t) = \lambda\,(y(t) - \cos(t))$ with initial value $y(0) = 1$. We march from $t = 0$ to $t = 1$ using one big step, and use $p = 10$ node points in the discretization. We only test real λ values for $\lambda = -10^k$, $k = 1, \cdots, 6$. Our numerical results show that the scheme using the uniform nodes converges at a faster rate compared to that using the



(a) (b)

Figure 2.19: Convergence rate for backward Euler preconditioned Gauss (a) and uniform (b) collocation formulations for different stiffness parameters λ.

Gauss nodes. For $\lambda = -1e+6$, the error decays rapidly when the uniform nodes are used. We therefore conclude that when the standard deferred correction scheme is preferred, the backward Euler preconditioned integral deferred correction schemes for the uniform collocation formulation are acceptable schemes to reduce the stiff error components. However order reduction (and divergence for large numbers of nodes) is still expected, e.g., the case when $\lambda = -1e+2$ in the numerical experiments.

2.4 Mapping Between Different Node Points

Analysis and numerical experiments in previous sections show that when the uniform nodes are used in the "convergence procedure", better convergence properties are usually expected compared with deferred correction schemes using other types of nodes. However the converged solutions are less accurate and may suffer from the Runge phenomenon. In this section, we show how to use different nodes for the provisional solution $\tilde{Y}$ and error δ for both the yp- and integral formulations so that when the deferred correction iterations for the uniform collocation formulations are convergent, the converged solution will solve the orthogonal basis based collocation formulations.

We first consider the yp-formulation given in Eq. (2.7); its error's equation is given by

$$\delta(t) = \left(f(t, y_0 + \int_0^t (\tilde{Y}(\tau) + \delta(\tau))d\tau) - f(t, y_0 + \int_0^t \tilde{Y}(\tau)d\tau) \right) + \varphi(t)$$

where $\varphi(t) = \left(f(t, y_0 + \int_0^t \tilde{Y}(\tau)d\tau) - \tilde{Y}(t)\right)$ is usually referred to as the residual function in the spectral deferred correction methods. Introducing the linear mapping P_{uG} which maps the polynomial values at the Gauss nodes to those at the uniform nodes, we can discretize the error's equation at uniform node points as

$$\tilde{\boldsymbol{\delta}}_u = \left(\mathbf{F}(\mathbf{t}_u, \mathbf{y}_0 + P_{uG}(\Delta t S_G \tilde{\mathbf{Y}}_G) + \Delta t \tilde{S_u}\tilde{\boldsymbol{\delta}}_u) - \mathbf{F}(\mathbf{t}_u, \mathbf{y}_0 + P_{uG}(\Delta t S_G \tilde{\mathbf{Y}}_G))\right) + P_{uG}\boldsymbol{\varphi}_G \qquad (2.37)$$

where $\boldsymbol{\varphi}_G = \mathbf{F}(\mathbf{t}_g, \mathbf{y}_0 + \Delta t S_G \tilde{\mathbf{Y}}_G) - \tilde{\mathbf{Y}}_G$ is the discretized residual at the Gauss collocation nodes, the sub-indices u and G represent that the corresponding vectors or integration matrices are defined on the uniform (u) or Gauss (G) nodes, respectively. Once the low order estimate of the error $\tilde{\boldsymbol{\delta}}_u$ is available, it can be mapped to the Gauss nodes using a precomputed linear mapping $P_{Gu} = P_{uG}^{-1}$; and $P_{Gu}\tilde{\boldsymbol{\delta}}_u$ can be added to the provisional solution $\tilde{\mathbf{Y}}_G$ defined on the Gauss nodes in the deferred correction procedure. Note that when the residual $\boldsymbol{\varphi}_G = \mathbf{0}$ (meaning that $\tilde{\mathbf{Y}}_G$ solves the Gauss collocation formulation), $\tilde{\boldsymbol{\delta}}_u = \mathbf{0}$. Similar to section 2.2.3, for a linear ODE of the form $y'(t) = \lambda y + f(t)$ with given initial condition $y(0) = y_0$, detailed matrix analysis shows that this mapping procedure, if applied from the beginning of the deferred correction iterations, is equivalent to solving the Gauss collocation formulation $\mathbf{Y}_G = \lambda(\mathbf{y}_0 + \Delta t S_G \mathbf{Y}_G) + \mathbf{F}_G$ (with given $\mathbf{y}_0 = [y_0, y_0, \cdots, y_0]^T$ and $\mathbf{F} = [f(t_1), f(t_2), \cdots, f(t_p)]^T$) using the preconditioner $P_{Gu}(I - \lambda\Delta t\tilde{S_u})^{-1}P_{uG}$. The preconditioned system is given by

$$P_{Gu}(I - \lambda\Delta t\tilde{S_u})^{-1}P_{uG}(I - \lambda\Delta t S_g)\mathbf{Y}_G = P_{Gu}(I - \lambda\Delta t\tilde{S_u})^{-1}P_{uG}(\lambda\mathbf{y}_0 + \mathbf{F}_G) = \mathbf{Y}^{[0]}. \qquad (2.38)$$

This mapping procedure can be applied in the same way to the integral Gauss collocation formulation represented by $\tilde{\mathbf{y}}_G = \mathbf{y}_0 + \Delta t S_G \mathbf{f}(\mathbf{t}_G, \tilde{\mathbf{y}}_G)$. Defining the residual function as $\boldsymbol{\varphi}_G = \mathbf{y}_0 + \Delta t S_G \mathbf{f}(\mathbf{t}_G, \tilde{\mathbf{y}}_G) - \tilde{\mathbf{y}}_G$, the discretized error's equation at uniform nodes is then given by

$$\tilde{\boldsymbol{\delta}}_u = \Delta t\tilde{S_u}\left(\mathbf{f}(\mathbf{t}_u, P_{uG}\tilde{\mathbf{y}}_G + \tilde{\boldsymbol{\delta}}_u) - \mathbf{f}(\mathbf{t}_u, P_{uG}\tilde{\mathbf{y}}_G)\right) + P_{uG}\boldsymbol{\varphi}_G, \qquad (2.39)$$

where the operators P_{Gu} and P_{uG} are the same operators as the yp-formulation. Clearly when $\boldsymbol{\varphi}_G = \mathbf{0}$, the error $\tilde{\boldsymbol{\delta}}_u = 0$.

CHAPTER 3

A New Jacobian-free Newton-Krylov Method

In this chapter, we will present a brief overview of general Jacobian-free Newton-Krylov (JFNK) methods [30]. Then, we will present a novel improved implementation of the JFNK algorithm made for deferred correction formulations. And finally, we will apply this novel technique to solve *stiff* time-dependent differential equations and compare performance results to general JFNK methods and SDC.

3.1 Krylov Methods

Before we present JFNK methods, it is important to review iterative methods to solve general matrix systems $Ax = b$. Solving $Ax = b$ using direct methods like Gaussian elimination is computationally expensive, requiring $O(N^3)$ operations. To avoid the cost of direct solvers, one can use iterative methods instead.

3.1.1 GMRES

One class of iterative methods is the Krylov methods, and one of the popular Krylov methods is the GMRES (generalized minimum residuals) method. We will now give a brief overview of GMRES. Note that a detailed overview of various Krylov methods can be found in [35, 36, 46, 51].

Assume we are trying to solve the linear system $Ax = b$ where $A \in \mathbb{C}^{n\times n}$ is a square matrix and $b \in \mathbb{C}^n$ is a vector. We will denote the Krylov subspace as $\mathcal{K}_m$ given by

$$\mathcal{K}_m(A, b) = span\left\{b, Ab, A^2b, \cdots, A^{m-1}b\right\}.$$

Assuming that A is nonsingular, GMRES is an iterative scheme that creates a series of converging solutions $x^{[m]}$ to the exact solution $x = A^{-1}b$. The idea of GMRES is to calculate an approximation $x^{[m]} \in \mathcal{K}_m$ that minimizes the L_2 norm (the reader should note that we denote $\|\cdot\|$ as the L_2 norm)

of the residual $r^{[m]} = Ax^{[m]} - b$. This turns out to be the same as solving a series of least squares problems to find $x^{[m]}$ [51].

GMRES starts by constructing a sequence of Krylov matrices Q_m whose orthogonal columns $q_1, q_2, \cdots, q_m$ span the successive Krylov subspace $\mathcal{K}_m$. During the m^{th} iteration of GMRES, GMRES attempts to solve the least squares problem

$$\|Ax^{[m]} - b\| = \text{minimum}. \tag{3.1}$$

We can express $x^{[m]}$ within the Krylov subspace $\mathcal{K}_m$ as $x^{[m]} = Q_m y^{[m]}$ where $y^{[m]} \in \mathbb{C}^n$ is a vector of coefficients for the the orthogonal basis $q_1 \cdots q_m$. The system now becomes

$$\|AQ_m y^{[m]} - b\| = \text{minimum}.$$

Note that applying Q^*_{m+1}, the conjugate transpose of Q_{m+1}, to the above equation does not change the value of the norm. Using this fact and the Hessenburg factorization $H_m = Q^*_{m+1} A Q_m$, we obtain

$$\begin{aligned} \|Q^*_{m+1} AQ_m y^{[m]} - Q^*_{m+1} b\| &= \text{minimum} \\ \|H_m y^{[m]} - \|b\|_2 e_1\| &= \text{minimum} \end{aligned} \tag{3.2}$$

where H_m is a $(m+1) \times m$ upper Hessenburg matrix and $e_1 = (1, 0, \cdots 0)^T$. At each iteration m of GMRES Eq. (3.2) is solved, and the approximate solution becomes $x^{[m]} = Q_m y^{[m]}$.

3.1.2 Inexact GMRES

The above GMRES description assumed that the matrix-vector product Av can be computed exactly. However, there are cases where one will not be able to have such a formulation. We now present the "*inexact*" GMRES method where we assume that the product Av *cannot* be computed exactly [15].

Assume we are trying to solve the linear system $Ax = b$. Let $x^{[0]}$ be an provisional solution and

$r^{[0]} = b - Ax^{[0]}$ the initial residual. Solving $Ax = b$ is equivalent to solving the residual equation

$$A\delta = r^{[0]} \tag{3.3}$$

where the exact solution $x = x^{[0]} + \delta$. Let $\delta^{[m]}$ be the approximate solution to Eq. (3.3) at the m^{th} iteration where $\delta^{[m]} \in \mathcal{K}_m(A, q_1)$. The Krylov subspace is given by

$$\mathcal{K}_m(A, q_1) = span\left\{q_1, Aq_1, A^2q_2, \cdots, A^{m-1}q_1\right\}$$

with $q_1 = \frac{r_0}{\|r_0\|}$. In addition, let $Q_{m+1} = [q_1, q_2, \cdots, q_{m+1}]$ be a matrix whose columns are the orthonormal basis of $\mathcal{K}_{m+1}(A, q_1)$. Q_{m+1} is generally obtained by the Arnoldi method (a Graham-Schmidt procedure applied to a Krylov subspace) [51]. We then write the *Arnoldi relation* as

$$AQ_m = Q_{m+1}H_m \tag{3.4}$$

where H_m is an $(m+1) \times m$ Hessenburg matrix.

In the inexact Krylov method, the matrix-vector product $Aq^{[m]}$ cannot be calculated exactly. Instead we have $(A + E_m)q_m$, where E_m is an error matrix. When this occurs, the Arnoldi relation Eq. (3.4) does not hold. Instead we have the *inexact Arnoldi relation*

$$\begin{aligned}[(A + E_1)q_1\,(A + E_2)q_2\,\cdots\,(A + E_m)q_m] &= Q_{m+1}H_m \\ AQ_m + [E_1q_1\,E_2q_2\,\cdots\,E_mq_m] &= Q_{m+1}H_m.\end{aligned} \tag{3.5}$$

Note that the above relation no longer spans the Krylov subspace $\mathcal{K}_{m+1}(A, q_1)$. However, as $\|E_iq_i\|$ decreases, we approach the true Arnoldi relation.

The exact GMRES method calculates the approximation $\delta^{[m]} = Q_m y^{[m]}$, where $y^{[m]}$ solves the least squares problem

$$\|H_m y^{[m]} - \|r^{[0]}\|e_1\| = \text{ minimum.} \tag{3.6}$$

In the inexact GMRES method, the approximation $\delta^{[m]}$ is computed in a similar manner using the matrices Q_{m+1} and H_m, satisfying the inexact Arnoldi relation Eq. (3.5). After $\delta^{[m]}$ is found, we

have two different types of residuals: the *computed* residual and the *true* residual. The computed residual $\tilde{r}^{[m]}$ is defined as

$$\begin{aligned}\tilde{r}^{[m]} &= b - Ax^{[m]} = b - A(x^{[0]} + \delta^{[m]}) \\ &= b - A(x^{[0]} + Q_m y^{[m]}) \\ &= r^{[0]} - AQ_m y^{[m]} \\ &= r^{[0]} - Q_{m+1} H_m y^{[m]}. \end{aligned} \tag{3.7}$$

Secondly, we will define the true residual $r^{[m]}$ as

$$\begin{aligned} r^{[m]} &= b - Ax^{[m]} \\ &= b - Ax^{[0]} - AQ_m y^{[m]} \\ &= r^{[0]} - AQ_m y^{[m]} \\ &= r^{[0]} - (Q_{m+1} H_m + [E_1 q_1 \, E_2 q_2 \, \cdots \, E_m q_m]) y^{[m]} \\ &= \tilde{r}^{[0]} - [E_1 q_1 \, E_2 q_2 \, \cdots \, E_m q_m] y^{[m]}. \end{aligned} \tag{3.8}$$

For the inexact GMRES method to converge to the exact GMRES method within a prescribed error tolerance $\epsilon > 0$, we must have $\|r^{[m]} - \tilde{r}^{[m]}\| \leq \epsilon$. The following theorem from [47] explains the conditions for convergence.

Theorem 3.1.1. *Let $\epsilon > 0$. Let $r^{[m]} = r^{[0]} - A\delta^{[m]}$, $\tilde{r}^{[m]} = r^{[0]} - Q_{m+1} H_m y^{[m]}$ be the true and computed residuals after m iterations of the inexact GMRES method, respectively with $y^{[m]}$ being the solution of Eq. (3.6). If for all $k = 1, \ldots, m$,*

$$\|E_k\| \leq \frac{\sigma_{min}(H_{m-1})}{m} \frac{\epsilon}{\|\tilde{r}_{k-1}\|} \tag{3.9}$$

where $\sigma_{min}(H_{m-1})$ is the smallest singular value of H_{m-1}, then $\|r^{[m]} - \tilde{r}^{[m]}\| \leq \epsilon$.

Note that the inexact GMRES assumes that $r^{[0]} = b - Ax^{[0]}$ is calculated exactly. If there is an error E_0, then the computed residual becomes $\tilde{r}^{[0]} = b - (A + E_0)x^{[0]} = r^{[0]} - E_0 x^{[0]}$. And the

inexact GMRES method converges to

$$\|\tilde{r}^{[0]} - A\delta^{[m]}\| = \text{ minimum.}$$

Hence, the inexact GMRES converges *not* to the solution

$$\|r^{[0]} - A\delta^{[m]}\| = \text{ minimum}$$

but to

$$\|r^{[0]} - A\delta^{[m]} - E_0 x^{[0]}\| = \text{ minimum.}$$

This may be a problem for overall convergence if $\|E_0\|$ is large [47].

3.2 General JFNK Methods

To begin, let's consider a general nonlinear algebraic system $M(x) = 0$ with N equations and N unknowns. Suppose an approximate solution $x^{[0]}$ is known. Newton's method can be used to iteratively compute convergent approximations

$$x^{[k+1]} = x^{[k]} + \delta x^{[k]}$$

where $k = 0, 1, \ldots$ For each iteration k, the vector $\delta x^{[k]}$ is found by solving the following system

$$J_M(x^{[k]})\delta x^{[k]} = b \tag{3.10}$$

where $b = -M(x^{[k]})$ and $J_M(x^{[k]})$ is the Jacobian matrix of $M(x)$ at $x^{[k]}$. We will call solving Eq. (3.10) solving the *Newton iteration*.

When the matrix $J_M(x^{[k]})$ is dense, computing the solution directly is computationally expensive, especially for large systems. For many special systems, the amount of operations necessary to find the solution can be significantly reduced. Consider systems with the following property

$$J_M(x^{[k]}) = \pm I - C$$

where most of the eigenvalues of the matrix C, λ, are such that $|\lambda| < 1$. Due to the rapid decay of eigenmodes in $C^m b$, instead of using Gaussian elimination, a more efficient approach is to iteratively search for the optimal solution in the Krylov subspace given by

$$\begin{aligned}\mathcal{K}_m(J_M, b) &= span\left\{b, (\pm I - C)b, (\pm I - C)^2 b, \cdots, (\pm I - C)^{m-1} b\right\} \\ &= span\left\{b, Cb, C^2 b, \cdots, C^{m-1} b\right\}.\end{aligned}$$

That is, in order to solve Eq. (3.10), we can use a typical Krylov method like GMRES to efficiently solve the Newton's iteration. When the Newton's method iterations and Krylov subspace methods are combined, they are referred to as the *Newton-Krylov methods*.

To have an efficient implementation of a Newton-Krylov method, the system to be solved needs the following properties:

1. A system $M(x)$ such that J_M is close to the identity matrix $\pm I$.

2. An efficient way to calculate the matrix-vector product Cb, or equivalently $J_M b$.

It is common that the original system $M(x)$ may not have favorable convergence properties for the Newton-Krylov method. A common technique to improve convergence is to apply a "preconditioner" to the original system such that the preconditioned system satisfies (1). In addition, it is usually the case the the Jacobian of the operator $M(x)$ is not easy to derive. To address point (2), a general forward difference approximation technique is used to approximate the Jacobian in most Newton-Krylov solvers. That is, we have $(\nabla_h M)(x) \approx J_M(x)$ where $(\nabla_h M)(x)$ is given by the following definition from [36].

Definition 3.2.1. Let $M(x)$ be defined in a neighborhood of $x \in \mathbb{R}^N$, $(\nabla_h M)(x)$ is the $N \times N$ matrix whose j^{th} column is given by

$$(\nabla_h M)(x)_j = \begin{cases} \frac{M(x + h\|x\|e_j) - M(x)}{h\|x\|} & \text{if } x \neq 0 \\ \frac{M(he_j) - M(x)}{h} & \text{if } x = 0 \end{cases}$$

for some parameter h.

We will denote using this Jacobian approximation technique within the Newton-Krylov method

as the *general Jacobian-free Newton-Krylov Method* (or general JFNK for short). A more detailed analysis of the choice of h can be found in [36].

3.3 Modified JFNK Method

In this section, we present the principles behind an improved JFNK method applied to deferred correction systems. We claim that by taking advantage of special properties of deferred correction systems, one is able to design a JFNK algorithm that will take far fewer calculations than general JFNK methods.

3.3.1 Deferred Correction Methods

Consider a general nonlinear algebraic system $A(y) = b$ with N equations and unknowns. In addition, assume given a provisional solution $\tilde{y}$, we have a method to approximate the error $\delta = y - \tilde{y}$ by

$$\tilde{\delta} = H(\tilde{y})$$

where $\tilde{y}$ is the input variable and the output variable, $\tilde{\delta}$, is an approximation to the error. Note that solving $A(y) = b$ is equivalent to solving $H(y) = 0$. Assume that we have a method that solves $H(y) = 0$ via a deferred correction formulation

$$\begin{cases} \delta^{[i]} & = H(y^{[i]}) \\ y^{[i+1]} & = y^{[i]} + \delta^{[i]} \ \ i = 1, 2, \ldots \end{cases}$$

that creates a series corrections $\delta^{[i]}$ that converge to a prescribed error tolerance. When the correction $\delta^{[i]}$ reaches the prescribed error tolerance, we will have $y^{[i+1]} = y^{[i]} + \delta^{[i]}$ as the solution to $A(y) = b$.

To accelerate deferred correction methods, we can view solving $H(y) = 0$ as a root finding problem. Hence, we can use Newton's method. Using $x^{[1]}$ as the provisional approximation within the Newton iteration, we obtain a series of converging solutions $x^{[k]}$ by using the following formulation

$$\begin{cases} J_H(x^{[k]})\delta x^{[k]} & = -H(x^{[k]}) \\ x^{[k+1]} & = x^{[k]} + \delta x^{[k]} \ \ k = 1, 2, \ldots \end{cases} \tag{3.11}$$

In addition, we will use the ideas behind Newton-Krylov methods. Namely, we assume that

$$J_H(\tilde{x}) = \pm I - C$$

where most of the eigenvalues of C, λ, are such that $|\lambda| < 1$.

3.3.2 Properties

Recall that in order for Newton-Krylov methods to be efficient, there must be a way to calculate $J_H v$ efficiently. Deferred correction algorithms have special properties that can be exploited to attain this goal. First, note that a deferred correction can be expressed as

$$\delta^{[m+1]} = H(y^{[m+1]}) = H(y^{[m]} + \delta^{[m]}).$$

Taylor expanding the above formulation, we can write an equation of the Jacobian matrix $J_H(y^{[m]})$ applied to the correction vector $\delta^{[m]}$ as

$$\delta^{[m+1]} - \delta^{[m]} = J_H(y^{[m]})\delta^{[m]} + O(\|\delta^{[m]}\|^2). \tag{3.12}$$

When the Jacobian $J_H(y^{[m]})$ is applied to an arbitrary correction vector $\delta^{[i]}$ where $i \neq m$, we can Taylor expand $J_H(y^{[m]})$ about $y^{[i]}$ to derive the following expression

$$\begin{aligned} J_H(y^{[m]})\delta^{[i]} &= J_H(y^{[i]})\delta^{[i]} + O(\|y^{[m]} - y^{[i]}\| \cdot \|\delta^{[i]}\|) \\ &= J_H(y^{[i]})\delta^{[i]} + O(\|\delta^{[i]}\| \sum_{j=i+1}^{m-1} \|\delta^{[j]}\|). \end{aligned}$$

Using Eq. (3.12), an expression of the Jacobian $J_H(y^{[m]})$ applied to an arbitrary correction vector $\delta^{[i]}$ when $i \neq m$ can be expressed as

$$J_H(y^{[m]})\delta^{[i]} = \delta^{[i+1]} - \delta^{[i]} + O(\|\delta^{[i]}\| \sum_{j=i}^{m-1} \|\delta^{[j]}\|). \tag{3.13}$$

3.3.3 The Krylov Subspace

For the vector $\delta^{[1]}$ and the Jacobian matrix $J_H = J_H(y^{[1]})$, the exact Krylov subspace is given by the following

$$\mathcal{K}_m(J_H, \delta^{[1]}) = span\left\{\delta^{[1]}, J_H\delta^{[1]}, J_H^2\delta^{[1]}, \cdots, J_H^{m-1}\delta^{[1]}\right\}.$$

Using Eq. (3.12) and Eq. (3.13), we can build an "inexact" Krylov subspace

$$\tilde{\mathcal{K}}_m(J_H, \delta^{[1]}) = span\left\{\delta^{[1]}, \delta^{[2]}, \delta^{[3]}, \cdots, \delta^{[m-1]}\right\} + \{0, \epsilon_2, \epsilon_3, \cdots, \epsilon_m\}$$

where the error terms $\epsilon_i = O(\|\delta^{[1]}\| \sum_{j=1}^{i-1} \|\delta^{[j]}\|)$ come from the approximations from Eq. (3.13). Note that this formulation resembles the inexact Arnoldi relation in Eq. (3.5) where $\epsilon_i = \|E_i\|$. Hence, the successive vectors $\delta^{[i]}$ from the deferred corrections formulation *approximately* span the Krylov subspace $\mathcal{K}_m(J_H, \delta^{[1]})$ in a similar manner of the inexact GMRES method.

3.3.4 Newton's Method

We will now lay the ground work of accelerating deferred correction methods via the Newton-Krylov framework by taking advantage of special properties when solving the Newton iterations

$$\begin{cases} J_H(x^{[k]})\delta x^{[k]} & = -H(x^{[k]}) \\ x^{[k+1]} & = x^{[k]} + \delta x^{[k]} \end{cases}$$

in solving $H(x) = 0$.

In terms of notation, we will designate the value $\delta_k^{[m]}$ as the m^{th} deferred correction iteration (also referred as an "inner" iteration) taking during the k^{th} Newton iteration (also referred as an "outer iteration"). Also, we will designate the approximate solutions $y_k^{[m]}$ to follow the same convention of notation.

First Newton Iteration To begin the Newton iteration, let the initial approximation in Newton's method can be provided by using the $(m+1)^{st}$ iterate $x_1^{[m+1]}$ of a deferred correction procedure.

That is, let $x^{[1]} = y_1^{[m+1]}$. We now obtain the following system to solve

$$J_H(y_1^{[m+1]})\delta x^{[1]} = -\delta_1^{[m+1]}. \tag{3.14}$$

Using $y_1^{[m+1]}$ instead of $y_1^{[1]}$ as an initial solution in Newton's method has two main properties that we will take advantage of. First, $y_1^{[m+1]}$ is closer to the solution than $y_1^{[1]}$, so we can expect Newton's method to converge more rapidly. Secondly, doing $m+1$ deferred correction iterations allows us to construct vectors that span the Krylov subspace $\tilde{\mathcal{K}}_m(J_H, \delta_1^{[1]})$.

To have efficiency of the Newton-Krylov method, assume $H(x)$ is preconditioned such that J_H has favorable convergence properties. Recall that for a Newton-Krylov method to be efficient there must be a way to calculate $J_H(\tilde{x})v$ efficiently. We will present a novel way to approximate the Jacobian without using a finite difference approximation.

If we express $\delta x^{[1]}$ in Eq. (3.14) as a linear combination of previous deferred correction vectors, which span $\tilde{\mathcal{K}}_m(J_H, \delta_1^{[1]})$, such that

$$\delta x^{[1]} = \sum_{i=1}^{m} c_i \delta_1^{[i]},$$

The Newton iteration, Eq. (3.14), can be rewritten as

$$J_H(y_1^{[m+1]}) \sum_{i=1}^{m} c_i \delta_1^{[i]} = -\delta_1^{[m+1]} \tag{3.15}$$

where the coefficients c_i are the unknowns. Applying the results from Eq. (3.13) to Eq. (3.15), an expression for the system to be solved, Eq. (3.14), may be written as

$$\sum_{i=1}^{m} c_i J_H(y_1^{[m+1]})\delta_1^{[i]} = -\delta_1^{[m+1]}$$

$$\sum_{i=1}^{m} \left[c_i(\delta_1^{[i+1]} - \delta_1^{[i]}) + O(\|\delta_1^{[i]}\| \sum_{j=i}^{m} \|\delta_1^{[j]}\|) \right] = -\delta_1^{[m+1]}.$$

Since the error should decrease with each deferred correction iteration, we should have $\|\delta_1^{[1]}\| =$

$\max_{1\le i\le m} \|\delta_1^{[i]}\|$. Hence, the above formulation becomes

$$\sum_{i=1}^{m} \left[c_i(\delta_1^{[i+1]} - \delta_1^{[i]}) + O(\sum_{j=i}^{m} \|\delta_1^{[1]}\|^2) \right] = -\delta_1^{[m+1]}.$$

$$\sum_{i=1}^{m} \left[c_i(\delta_1^{[i+1]} - \delta_1^{[i]}) + O\left((m-i+1)\|\delta_1^{[1]}\|^2\right) \right] = -\delta_1^{[m+1]}.$$

$$\sum_{i=1}^{m} c_i(\delta_1^{[i+1]} - \delta_1^{[i]}) + O\left(\frac{(m+1)m}{2}\|\delta_1^{[1]}\|^2\right) = -\delta_1^{[m+1]}. \tag{3.16}$$

Instead of solving Eq. (3.14) for $\delta x^{[1]}$, one can efficiently solve the following preconditioned system

$$\sum_{i=1}^{m} c_i(\delta_1^{[i+1]} - \delta_1^{[i]}) = -\delta_1^{[m+1]} \tag{3.17}$$

where $\mathbf{c}$, a vector consisting of the coefficients c_i, is the unknown. Note that from Eq. (3.16) the solution of the preconditioned system $\delta x^{[1]} = \sum_{i=1}^{m} c_i \delta_1^{[i]}$ satisfies Eq. (3.14) up to $O\left(\frac{(m-1)m}{2}\|\delta_1^{[1]}\|^2\right)$.

Recall that $x^{[k]}$ is the solution after k Newton iterations. After solving Eq. (3.17), one updates the solution used in the Newton iteration $x^{[1]} = y_1^{[m+1]}$ with $x^{[2]} = y_1^{[m+1]} + \delta x^{[1]}$. Let $\delta_2^{[1]} = H(x^{[2]})$; we can approximate the magnitude $\|\delta_2^{[1]}\|$ using the formula

$$\begin{aligned}\|\delta_2^{[1]}\| &= \|H(y_1^{[m+1]} + \delta x^{[1]})\| \\ &= \|\delta_1^{[m+1]} + J_H(y_1^{[m+1]})\delta x^{[1]} + O(\|\delta x^{[1]}\|^2)\|.\end{aligned}$$

From Eq. (3.16), we obtain the following estimate of $\|\delta_2^{[1]}\|$,

$$\|\delta_2^{[1]}\| = O\left(\frac{(m+1)m}{2}\|\delta_1^{[1]}\|^2 + \|\delta x^{[1]}\|^2\right).$$

Notice that we can expect $\|\delta x^{[1]}\| = O(\|\delta_1^{[1]}\|)$ since $\delta x^{[1]} = \sum_{i=1}^{m} c_i \delta_1^{[i]}$. We obtain a sizeable drop or "jump" in the error since $\|\delta_2^{[1]}\| \ll \|\delta_1^{[m+1]}\|$. If $\|\delta_2^{[1]}\| < \epsilon$ where ϵ is a prescribed tolerance, we have converged; and we stop the Newton's method. The approximation to the solution is $x^{[2]} = y_1^{[m+1]} + \delta x^{[1]}$.

Consider the case when $\|\delta_2^{[1]}\|$ has not converged to a prescribed error such that $\|\delta_2^{[1]}\| \ge \epsilon$. We

will we attempt to solve the Newton iteration

$$J_H(x^{[2]})\delta x^{[2]} = -\delta_2^{[1]}$$

where $\delta x^{[2]}$ is a linear combination of m correction vectors.

We want to update the Krylov subspace $\tilde{\mathcal{K}}_m$ by using the latest information. Therefore, we will add new Krylov vector and remove $\|\delta_1^{[1]}\|$ from the Krylov subspace $\tilde{\mathcal{K}}_m$. In order for a vector the span the subspace, the m correction vectors must be such that the approximation in Eq. (3.13) holds. Keeping this in mind, one can see that $\delta x^{[1]}$ cannot be included within the subspace due to two reasons.

1. The correction vectors in $\delta x^{[1]} = \sum_{i=1}^{m} c_i \delta_1^{[i]}$ already span the original Krylov subspace $\tilde{\mathcal{K}}_m(J_H, \delta_1^{[1]})$ where $J_H = J_H(y_1^{[m+1]})$. Hence, $\delta x^{[1]}$ does not provide new information in terms of the Krylov subspace.

2. The jump in the error $\|\delta_2^{[1]}\|$ may cause the assumption from Eq. (3.13) to not hold. To see this, let's consider

$$\begin{aligned}\delta_2^{[1]} &= H(x^{[2]}) = H(y_1^{[m+1]} + \delta x^{[1]}) \\ &= \delta_1^{[m+1]} + J_H(y_1^{[m+1]})\delta x^{[1]} + O(\|\delta x^{[1]}\|^2).\end{aligned}$$

   Writing $J_H(y_1^{[m+1]})$ as $-I - C$, we obtain

$$\delta_2^{[1]} = \delta_1^{[m+1]} + (-I - C)\delta x^{[1]} + O(\|\delta x^{[1]}\|^2).$$

   If the eigenvalues of the matrix C, λ, are such that $J_H \not\approx -I$, our Jacobian matrix-vector product approximation from Eq. (3.13) will not be valid. Under these conditions, the magnitude of the right hand side of the above equation is $O(\|\delta_1^{[1]}\|)$ while $\|\delta_2^{[1]}\|$ is $O(\|\delta_1^{[1]}\|^2)$. Note that this phenomenon may be observed when solving stiff ODE systems with deferred correction methods like SDC.

Due to the above reasons, updating the Krylov subspace calls for adding the vectors $\delta_2^{[i]}$. In fact, we have two different strategies to consider for updating the Krylov subspace for the k^{th} Newton

iteration:

1. **The restart strategy.** This idea is popular in "restarted" GMRES procedures. Given $\delta_k^{[1]}$, one can do an additional m deferred correction iterations to use $y_k^{[m+1]}$ as the provisional solution for the Newton iteration and use $\delta_k^{[i]}$ $i = 1, \ldots, m$ vectors to span $\tilde{\mathcal{K}}_m(J_H, \delta_k^{[1]})$ where $J_H = J_H(y_k^{[m+1]})$. Afterwards, we can use m of the new deferred corrections in $\delta x^{[k]} = \sum_{i=1}^{m} c_i \delta_k^{[i]}$ and solve Eq. (3.14) by using the novel JFNK strategy. This is equivalent of restarting the entire deferred correction JFNK procedure.

2. **The recycle strategy.** One can calculate $p < m$ new deferred correction vectors and remove the first p deferred correction vectors in $\delta x^{[k-1]}$ while keeping vectors $\delta_{k-1}^{[p+1]}, \cdots, \delta_{k-1}^{[m]}$, hence recycling them, and adding the new p deferred correction vectors. Hence we solve Eq. (3.14) by using
$$\delta x^{[k]} = c_1 \delta_{k-1}^{[p+1]} + \cdots + c_{m-p} \delta_{k-1}^{[m]} + c_{m-p+1} \delta_k^{[1]} + \cdots c_m \delta_k^{[p]}.$$

In the next section, we take into account of the properties discussed so far when providing the initial design of the modified JFNK algorithm. In the algorithm, we will focus on the restart strategy and leave the recycle strategy for future research.

3.4 Algorithm Design

Assume we are trying to solve the deferred correction $H(y) = 0$. In the algorithm, we will continue using the deferred correction type function evaluations $\tilde{\boldsymbol{\delta}}^{[k]} = H(\tilde{\mathbf{y}}^{[k-1]} + \tilde{\boldsymbol{\delta}}^{[k-1]})$ as they effectively control the growth of the non-stiff errors, even though the Jacobian matrix of the low order techniques preconditioned system is no longer close to $-I$ for the stiff components.

We introduce a predefined but adjustable parameter $\eta_1 < 1$ to check if the initial provisional solution provided by the predictor can serve as a good initial guess for the Newton's method when solving the nonlinear collocation formulation. We also include another parameter $\eta_2 < 1$ to check if the standard deferred correction schemes are still effective. When order reduction or divergence is observed, we search for the optimal solution in the Krylov subspace, $\tilde{K}_m$, using a modified Jacobian-free Newton-Krylov method. The Krylov subspace is updated when the low order estimate $\tilde{\boldsymbol{\delta}}^{[k]} = H(\tilde{\mathbf{y}}^{[k-1]} + \tilde{\boldsymbol{\delta}}^{[k-1]})$ shows no significant improvement compared with previous step

results. When the subspace $\mathcal{K}_m$ is complete with m vectors, the optimal solution for the linearized equation $J_H \delta x = -\tilde{\boldsymbol{\delta}}^{[k]}$ in each Newton's iteration is sought in the updated Krylov subspace. In the modified JFNK, instead of the finite difference approximation as used in standard JFNK methods, the matrix-vector product $J_H \tilde{\boldsymbol{\delta}}^{[k-1]}$ is computed using the Taylor expansion

$$\tilde{\boldsymbol{\delta}}^{[k]} = H(\tilde{\mathbf{y}}^{[k-1]} + \tilde{\boldsymbol{\delta}}^{[k-1]}) \approx \tilde{\boldsymbol{\delta}}^{[k-1]} + J_H \tilde{\boldsymbol{\delta}}^{[k-1]},$$

which is valid when $O(||\tilde{\boldsymbol{\delta}}^{[k]}||) \approx O(||\tilde{\boldsymbol{\delta}}^{[k-1]}||)$, i.e., when the result from one deferred correction iteration no longer converges efficiently for stiff systems. We stop the iterations in the "convergence procedure" when the solution is sufficiently close to that of the collocation formulation, measured by a prescribed error tolerance. The algorithm is described in detail by the following pseudo-code.

JFNK based "convergence procedure"

Step 1: Predictor: Use a "good" low order method to find an approximate solution $\tilde{\mathbf{y}}^{[0]}$

Step 2: Check $\tilde{\mathbf{y}}^{[0]}$: Use a "good" low order method to solve the error's equation to get a low order estimate of the error $\tilde{\boldsymbol{\delta}}^{[0]} = H(\tilde{\mathbf{y}}^{[0]})$.

if $||\tilde{\boldsymbol{\delta}}^{[0]}||/||\tilde{\mathbf{y}}^{[0]}|| < \eta_1$,

$\tilde{\mathbf{y}}^{[1]} = \tilde{\mathbf{y}}^{[0]} + \tilde{\boldsymbol{\delta}}^{[0]}$ continue

else

Select a smaller time step size, go to Step 1.

endif

Step 3: Standard Deferred Correction Iterations: Start from $k = 1$, update the error's equation and get a low order estimate of the error $\tilde{\boldsymbol{\delta}}^{[k]} = H(\tilde{\mathbf{y}}^{[k]})$.

if $||\tilde{\boldsymbol{\delta}}^{[k]}|| < \epsilon$,

Go to Step 5 with the converged solution $\tilde{\mathbf{y}}^{[k]} + \tilde{\boldsymbol{\delta}}^{[k]}$.

elseif $||\tilde{\boldsymbol{\delta}}^{[k]}||/||\tilde{\boldsymbol{\delta}}^{[k-1]}|| < \eta_2$,

$\tilde{\mathbf{y}}^{[k+1]} = \tilde{\mathbf{y}}^{[k]} + \tilde{\boldsymbol{\delta}}^{[k]}$, $k++$, repeat Step 3.

else

Set $k = 1$. Add $\tilde{\boldsymbol{\delta}}^{[1]}$ as the first vector in the Krylov subspace.

Go to Step 4.

endif

Step 4: Modified JFNK:

if $k < m + 1$

Update $\tilde{\mathbf{y}}^{[k+1]} = \tilde{\mathbf{y}}^{[k]} + \tilde{\boldsymbol{\delta}}^{[k]}$.

Evaluate $\tilde{\boldsymbol{\delta}}^{[k+1]} = H(\tilde{\mathbf{y}}^{[k+1]})$.

if $||\tilde{\boldsymbol{\delta}}^{[k+1]}|| < \epsilon$,

Go to Step 5 with the converged solution $\tilde{\mathbf{y}}^{[k+1]}$.

else

Update the Krylov subspace, by adding $\tilde{\boldsymbol{\delta}}^{[k+1]}$ and updating the corresponding $J_H\tilde{\boldsymbol{\delta}}^{[k]}$, and by removing any outdated (inaccurate) $\tilde{\boldsymbol{\delta}}^{[j]}$ and $J_H\tilde{\boldsymbol{\delta}}^{[j]}$.

$k++$, go to Step 4.

endif

Solve the linearized equation $J_H(\tilde{\mathbf{y}}^{[m+1]})\delta x = -\tilde{\boldsymbol{\delta}}^{[m+1]}$ by searching for the optimal solution in the Krylov subspace $\tilde{\mathcal{K}}_m(J_H, \tilde{\boldsymbol{\delta}}^{[1]})$.

Set $\tilde{\mathbf{y}}^{[0]} = \tilde{\mathbf{y}}^{[m+1]}$, $\tilde{\boldsymbol{\delta}}^{[0]} = \delta x$.

Set $k = 0$, go to Step 4.

Step 5: Output: Output the computed approximate solution.

3.5 Numerical Experiments

In this section, we show some preliminary numerical results in solving linear and nonlinear *stiff* ODE systems using the following techniques: a spectral deferred correction method, the general JFNK, and the modified JFNK. We compare their performance results over the coarse of one time interval of size Δt. Each method attempts to find a solution to the deferred correction formulation $H(y) = 0$. The deferred correction formulation $\tilde{\delta} = H(\tilde{y})$ that we use is the backward Euler, integral formulation of SDC consisting of 10 Gauss-Lobatto nodes. Recall that converged solution is not the exact solution to the ODE system but the Gauss collocation solution Eq. (2.4).

For each example, the predictor solution is gained via the backward Euler method. The modified JFNK method is implemented as follows. Note that the we relate the following procedure to the

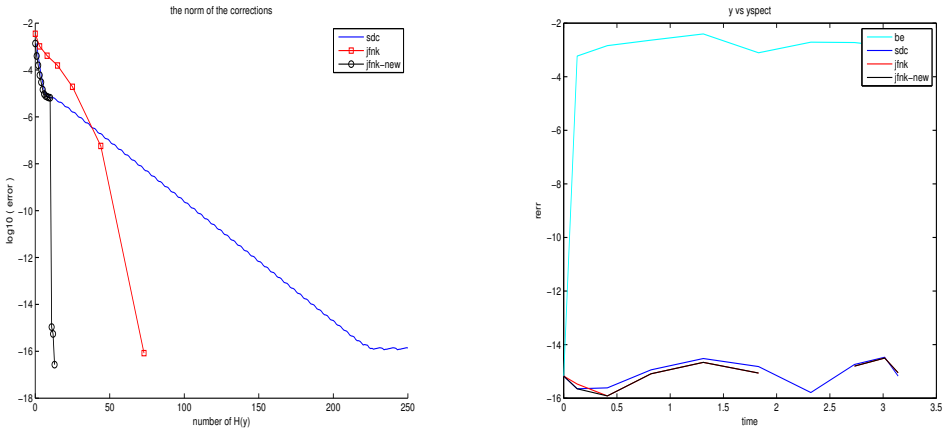
steps from the proposed pseudo-code from the previous section. After obtaining the provisional solution, the deferred correction JFNK does $k_0 \geq 0$ additional iterations to obtain an adequate low order approximation (Step 3). For the k^{th} Newton iteration, 11 SDC iterations are used to gain the $\delta_k^{[i]}$ $i = 1, \ldots, 11$ used in solving the Newton iteration $\delta x^{[k]} = \sum_{i=1}^{10} c_i \delta_k^{[i]}$ with the initial solution for the Newton iteration as $x^{[k]} = y_k^{[11]}$ in conjunction with Eq. (3.11) (Step 4). Lastly, note that the number of SDC iterations 11 corresponds to the number of Lobatto nodes + 1.

3.5.1 Cosine Problem

The system that we are trying to solve is

$$\begin{cases} (y(t) - p(t))' & = \lambda(y(t) - p(t)) \\ y(0) & = p(0) \end{cases}$$

where $p(t) = \cos(t)$. The exact solution is $y(t) = p(t)$. When λ is a large negative number, the system is stiff. We set $\lambda = -100$ and $\Delta t = \pi$ and solve the system within $t \in [0, \Delta t]$. We show the results in figure 3.1.



(a) (b)

Figure 3.1: Cosine problem. (a): the magnitude of the deferred correction $\frac{\|\tilde{\delta}\|}{\|\tilde{y}\|}$. (b): the relative error of the final iteration vs. the collocation formulation

The results from figure 3.1 (a) show the relative magnitude of the calculated deferred correction from the following methods: SDC (sdc), general JFNK (jfnk), and the modified JFNK (jfnk-new).

First thing to note is that the SDC method suffers from order reduction as predicted. This leads to a high number of SDC iterations needed to converge. The general JFNK method is able to converge must faster than SDC. We can see that the modified JFNK method ($k_0 = 0$) is in line with SDC for the first 11 iterations. Then there is a large jump in the error due to solving Eq. (3.11). Since we are solving a single-mode linear problem, the approximation Eq. (3.12) is exact and explains why after the Newton iteration we practically converge to the solution.

The results from figure 3.1 (b) show the relative error of the backward Euler provisional solution (be), SDC, general JFNK (jfnk), and modified JFNK (jfnk-new) methods compared to the collocation formulation. All three methods are able to converge to the collocation formulation.

3.5.2 Linear Multimode Problem

The system that we are trying to solve is

$$\begin{cases} (\bar{y}(t) - \bar{p}(t))' & = B(\bar{y}(t) - \bar{p}(t)) \\ \bar{y}(0) & = \bar{p}(0). \end{cases}$$

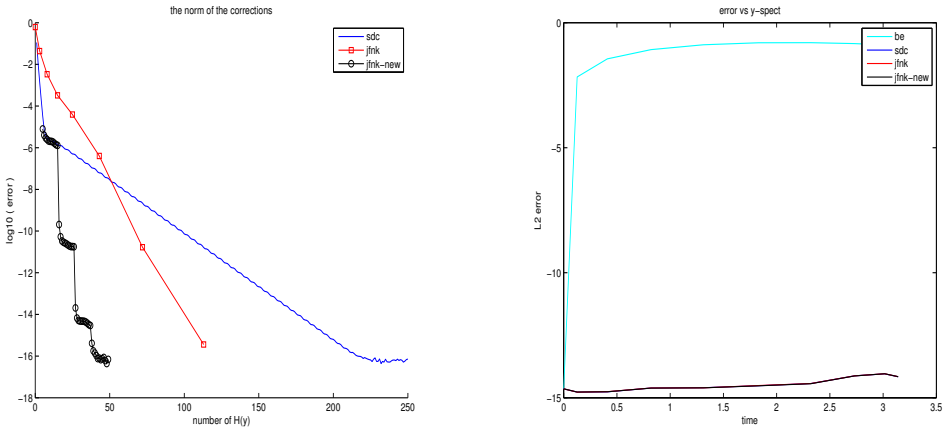
where $\bar{y}(t)$ and $\bar{p}(t)$ are vectors of dimension M. The exact solution is $\bar{y}(t) = \bar{p}(t)$. The matrix B is constructed by

$$B = U^T \Lambda U,$$

where U is a randomly generated orthogonal matrix, Λ is a diagonal matrix whose diagonal entries $\{\lambda_i\}_{i=1}^{M}$ are all negative. We have $\bar{p}_i(t) = \cos(t + \alpha_i)$ with $\alpha_i = 2\pi i/M$.

For this experiment, $M = 7$ and $\lambda = [-100\ -1\ -1\ -1\ -1\ -1\ -1]$. We set $\Delta t = \pi$ and model the system from $t \in [0, \Delta t]$. Figure 3.2 shows the numerical results.

The results from figure 3.2 (a) show that SDC suffers from order reduction due to the stiffness of the problem. The general JFNK converges faster than SDC, but takes more iterations than in the single mode cosine problem. Since the initial SDC iterations converge quickly, we start the deferred correction JFNK method's Newton iterations Eq. (3.11) after $k_0 = 5$ iterations. Recall that in order for Eq. (3.12) to be valid, $\|\delta^{[k+1]}\| \ll \|\delta^{[k]}\|$. That is, if the error of successive iterates drops too fast, we cannot add those correction vectors to the Krylov subspace. Instead, we do $k0 = 5$ SDC



(a) (b)

Figure 3.2: Linear multimode problem. (a): the magnitude of the deferred correction $\frac{\|\tilde{\delta}\|}{\|\tilde{y}\|}$. (b): the relative error of the final iteration vs. the collocation formulation

iterations so that order reduction can slow down the convergence of SDC.

Although the problem is linear, the deferred correction JFNK undergoes a series of Newton iterations, each causing a jump in the error. This occurs because the problem is a multimode problem, unlike the cosine problem, which is single mode. Nevertheless, we see that the deferred correction JFNK outperforms SDC and the general JFNK.

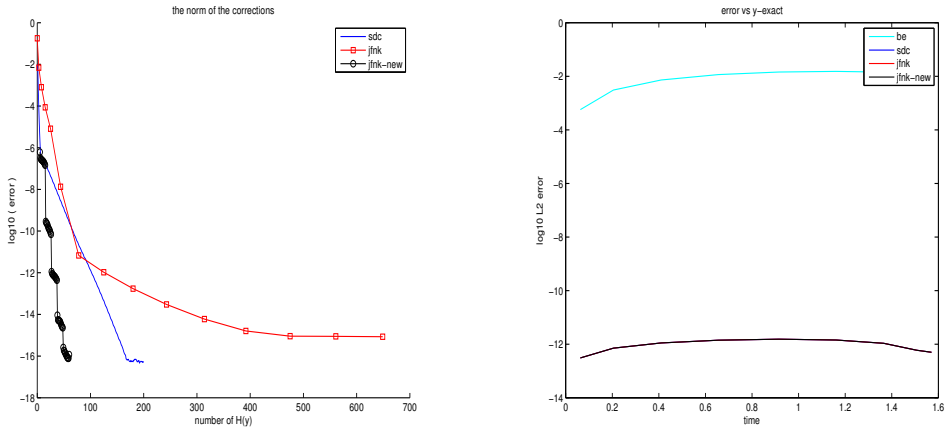
3.5.3 Nonlinear Multimode Problem

The system that we are trying to solve is

$$\begin{cases} (y_i(t) - p_i(t))' & = \lambda y_{i+1}(t)(y_i(t) - p_i(t)),\ 1 < i < M-1 \\ (y'_M(t) - p_M(t))' & = \lambda(y_i(t) - p_i(t)),\ i = M. \end{cases}$$

The exact solution is $y_i(t) = p_i(t)$ where $p_i(t) = 2 + \cos(t + \alpha_i)$ and $\alpha_i = 2\pi i/M$. For the numerical simulation, we set $M = 7$, $\lambda = [-100\ -100\ -1\ -1\ -1\ -1\ -1]$, $\Delta t = \pi/2$, and $t \in [0, \Delta t]$. Figure 3.3 shows the numerical results.

The results from figure 3.3 show the SDC method suffering from order reduction, once again. However, the general JFNK requires does not converge to the accuracy of SDC. The modified JFNK with $k_0 = 5$ is able to converge much faster than the other two methods since it needs 50 iterations.



(a) (b)

Figure 3.3: Nonlinear multimode problem. (a): the magnitude of the deferred correction $\frac{\|\tilde{\delta}\|}{\|\tilde{y}\|}$. (b): the relative error of the final iteration vs. exact solution

3.5.4 Van der Pol Oscillator

This last example describes the behavior of vacuum tube circuits. The following formulation was proposed by B. Van der Pol in the 1920's and is referenced as the Van Der Pol Oscillator [29]. The system that we are trying to solve is

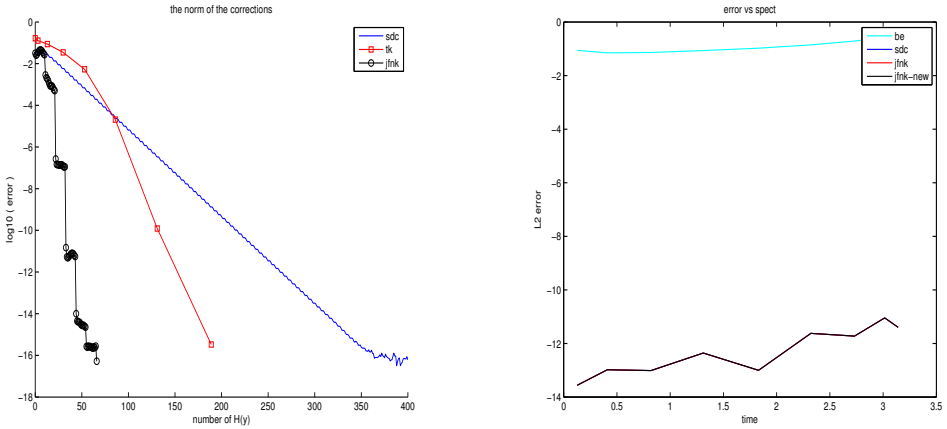
$$\begin{cases} y_1'(t) &= y_2(t) \\ y_2'(t) &= \lambda(-1 + y_1^2(t)y_2(t) + y_1(t)). \end{cases}$$

where the initial condition is $[y(0), y'(0)] = [y_1(0), y_2(0)] = [2, -0.6666654321121172]$.

For this experiment, $\lambda = -100$. We set $\Delta t = \pi$ and model the system from $t \in [0, \Delta t]$. Figure 3.4 shows the numerical results of solving this system.

The results from figure 3.4 show that the SDC method suffers from order reduction. The general JFNK is able to converge faster than SDC. The modified JFNK with $k_0 = 0$ is able to converge much faster than the other two methods.

We summarize the results from the the numerical experiments in Table 3.1 and Table 3.2. We see that for stiff systems, SDC suffers from order reduction and does not perform well. It is interesting to note that the general JFNK outperforms SDC in all examples except the nonlinear multimode problem. Nevertheless, the modified JFNK method is able to converge much faster than SDC and



(a) (b)

Figure 3.4: Van der Pol oscillator. (a): the magnitude of the deferred correction $\frac{\|\tilde{\delta}\|}{\|\tilde{y}\|}$. (b): the relative error of the final iteration vs. the collocation formulation

problem	SDC	general JFNK	modified JFNK
cosine	255	73	13
linear multimode	220	113	42
nonlinear multimode	180	649	50
Van der Pol	360	189	66

Table 3.1: The number of $H(\tilde{\mathbf{y}})$ needed to converge.

problem	SDC	general JFNK	modified JFNK
cosine	-15.9	-16.0	-16.6
linear multimode	-16.1	-15.4	-16.2
nonlinear multimode	-15.9	-15.1	-15.7
Van der Pol	-16.2	-15.5	-16.3

Table 3.2: The relative correction $\log_{10}\left(\frac{\|\tilde{\delta}\|}{\|\tilde{y}\|}\right)$ of the converged solution.

is about 3 to 4 times faster than the general JFNK method.

CHAPTER 4

Parallel Full Approximation Scheme in Space and Time

There is new interest in developing numerical algorithms with parallel temporal integration. The creation of the parareal algorithm in 2001 by Lions, Maday, and Turinici has given insight on making numerical parallel integration methods for solving ordinary differential equations (ODEs) and partial differential equations (PDEs) [42]. Parareal is an iterative method where the algorithm's parallel efficiency is bounded by $\frac{1}{K}$ where K is the number of parareal iterations needed to converge. To address this limitation, Minion and Emmett created another parallel integration method called the parallel full approximation scheme in space and time, known as PFASST [17]. PFASST is an iterative method the combines ideas from parareal, spectral deferred corrections (SDC), multigrid, and the full approximation scheme (FAS). PFASST's parallel efficiency is bounded by $\frac{K_s}{K_p}$ where K_p and K_S are the number of iterations need for PFASST and serial SDC to converge respectively [17]. This chapter will review the temporal and spatial properties that lie within PFASST.

4.1 Temporal Methods

In this section we will describe the numerical time stepping methods used for parallel integration in time. First, we will briefly mention some favorable properties of SDC. Afterwards, we will review a parallel integration method called *parareal* followed by the *full approximation scheme* (FAS), popular in multigrid methods. And finally, we will see how all of these methods merge to form the parallel time-stepping algorithm, the parallel full approximation scheme in space and time (PFASST).

4.1.1 Spectral Deferred Corrections

PFASST uses SDC as a main component of its temporal integration scheme. SDC has many favorable properties to be considered when integrating in time. Other than being able to converge

to the solution of the Gaussian collocation formulation, SDC's use of low order methods allows construction of methods that use operator splitting and/or multirate time-stepping [7, 8, 39]. In addition, SDC can be modified to create semi-implicit or IMEX schemes [17]. To see this, assume we are trying to solve the following ODE system

$$\begin{cases} y'(t) = f_E(t, y(t)) + f_I(t, y(t)) \\ y(0) = y_0 \end{cases}$$

where the first term on the right-hand side f_E should be treated explicitly and the second term f_I should be treated implicitly. The SDC sweep becomes

$$\begin{aligned} \tilde{y}_{m+1}^{[k+1]} = \tilde{y}_m^{[k+1]} &+ \Delta t_m [f_E(t_m, \tilde{y}_m^{[k]} + \tilde{\delta}_m) - f_E(t_m, \tilde{y}_m^{[k]})] \\ &+ \Delta t_m [f_I(t_{m+1}, \tilde{y}_{m+1}^{[k]} \tilde{\delta}_{m+1}) - f_I(t_{m+1}, \tilde{y}_{m+1}^{[k]})] + S_m^{m+1} f(t, \tilde{y}^{[k]}) - \tilde{y}_{m+1}^{[k]} + \tilde{y}_m^{[k]}. \end{aligned}$$

where $\tilde{y}_m^{[k]}$ is the k^{th} iteration's approximation of $y(t_m)$, $\tilde{y}^{[k]}$ is a vector consisting of all the values of $\tilde{y}_m^{[k]}$, and $S_m^{m+1} f(t, \tilde{y}^{[k]})$ is the spectral integration matrix applied to $f(t, \tilde{y}^{[k]})$ from $t \in [t_m, t_{m+1}]$.

4.1.2 Parareal

The second component of the PFASST algorithm is the parareal method. The following is a brief explanation of the parareal method. Assume we are solving the general ODE system

$$\begin{cases} y'(t) &= f(t, y(t)) \\ y(0) &= y_0 \end{cases}$$

where $t \in [0, T_N]$. The domain is divided into N intervals where $T_n = n\Delta t$ and $\Delta t = \frac{T_N}{N}$. Parareal first computes a provisional solution Y_n using a low-order method (like backward Euler) in *serial* over the entire time domain $[0, T_N]$

$$\begin{cases} \frac{Y_{n+1} - Y_n}{\Delta t} &= f(T_{n+1}, Y_{n+1}) \\ Y_0 &= y_0 \end{cases}$$

for $n = 1 \cdots N$. We will call Y_n the "coarse solution." Next, we use the calculated solutions Y_n as initial conditions for each interval $[T_n, T_{n+1}]$ and solve the following system *exactly* for $y_n(t)$ in the interval $[T_n, T_{n+1}]$

$$\begin{cases} y_n'(t) & = f(t, y_n(t)) \\ y_n(T_n) & = Y_n. \end{cases}$$

We will call $y_n(t)$ the "fine solution." In practice, it is impossible to solve the above system exactly; instead, one uses a highly accurate solve. The above system can be solved in *parallel* if each processor $\mathbf{P}_n$ is responsible for solving for $y_n(t)$, where $t \in [T_n, T_{n+1}]$. Parareal then approximates the error $\delta(t) = y(t) - y_n(t)$ in the interval $t \in (T_n, T_{n+1}]$ by solving an error's correction. The error's equation can be written as

$$\begin{aligned} \delta(t) &= y(t) - y_n(t) \\ &= y(T_n) + \int_{T_n}^{t} f(\tau, y(\tau))\,\mathrm{d}\tau - Y_n - \int_{T_n}^{t} f(\tau, y_n(\tau))\,\mathrm{d}\tau \\ &= y(T_n) - Y_n + \int_{T_n}^{t} f\left(\tau, y_n(\tau) + \delta(\tau)\right) - f\left(\tau, y_n(\tau)\right)\,\mathrm{d}\tau. \end{aligned} \tag{4.1}$$

Using Eq. (4.1), an expression for $\delta(T_{n+1})$ can be written as

$$\delta(T_{n+1}) = y(T_n) - Y_n + \int_{T_n}^{T_{n+1}} f\left(\tau, y_n(\tau) + \delta(\tau)\right) - f\left(\tau, y_n(\tau)\right)\,\mathrm{d}\tau.$$

We can calculate an approximation of the correction $\tilde{\delta}$ by using a low-order method (like backward Euler) to approximate the integral. In addition, since $y(t)$ is the unknown, we can approximate $y(T_n)$ in the above formulation as $y(T_n) \approx y_{n-1}(T_n) + \tilde{\delta}(T_n)$. Keeping this in mind, we can write an expression for the correction as

$$\tilde{\delta}_{n+1} = \tilde{\delta}_n + y_{n-1}(T_n) - Y_n + \Delta t[f(T_{n+1}, y_n(T_{n+1}) + \tilde{\delta}_{n+1}) - f\left(T_{n+1}, y_n(T_{n+1})\right)].$$

Finally we finish the "parareal iteration" by updating the coarse solutions $Y_n^{[1]} = y_{n-1}(T_n) + \tilde{\delta}_n$ in *serial*. The process continues with calculating a new fine solution. That is, assuming we have already done k iterations of parareal, the coarse solution is updated $Y_n^{[k+1]} = y_{n-1}^{[k]}(T_n) + \tilde{\delta}_n^{[k]}$. Parareal iterates by using $Y_n^{[k+1]}$ as the new initial condition when calculating the fine solution in

$t \in (T_n, T_{n+1}]$

$$\begin{cases} \frac{d}{dt} y_n^{[k+1]}(t) & = f(t, y_n^{[k+1]}(t)) \\ y_n^{[k+1]}(T_n) & = Y_n^{[k+1]} \end{cases}$$

and so on.

Parareal can also be understood in terms of two numerical approximation methods: $\mathcal{G}$ and $\mathcal{F}$. Both $\mathcal{G}$ and $\mathcal{F}$ are ODE methods that take an initial condition $\tilde{y}_n \approx y(T_n)$ and compute the solution to the ODE from T_n to T_{n+1}. For parareal to be efficient, $\mathcal{G}$ must be computationally less expensive than $\mathcal{F}$. Hence, $\mathcal{G}$ is usually a low-order method; and $\mathcal{F}$ is a high-order method. In addition, $\mathcal{F}$ also may use a smaller time step than $\mathcal{G}$ because the accuracy of parareal is limited by the accuracy of $\mathcal{F}$. Thus, we will refer to $\mathcal{G}$ as the *coarse propagator* and $\mathcal{F}$ as the *fine propagator* [17].

Parareal begins by calculating calculating the coarse solution in serial using the coarse propagator

$$Y_{n+1}^{[k]} = \mathcal{G}(T_{n+1}, T_n, Y_n^{[k]})$$

for $n = 0 \cdots N-1$ with $y_0 = y(0)$. These values act as the initial conditions for the respective processor in the parallel computation

$$y_{n+1}^{[k]} = \mathcal{F}(T_{n+1}, T_n, Y_n^{[k]}).$$

Parareal continues iteratively by alternating between the parallel computation of $\mathcal{F}(T_{n+1}, T_n, Y_n^{[k]})$ and updating the initial conditions at each processor of the form

$$Y_{n+1}^{[k+1]} = \mathcal{G}(T_{n+1}, T_n, Y_n^{[k+1]}) + \mathcal{F}(T_{n+1}, T_n, Y_n^{[k]}) - \mathcal{G}(T_{n+1}, T_n, Y_n^{[k]}) \tag{4.2}$$

for $n = 0 \cdots N-1$ with $y_0 = y(0)$ [17].

After N iterations of the parareal method, the solution is equal to applying $\mathcal{F}$ in serial. In practice, the iterations converge much more quickly for large N. Parareal provides speedup if the number of iterations needed to converge K is significantly less than N. Recall that the parallel efficiency in parareal is bounded by $\frac{1}{K}$, which limits parareal's effectiveness and scalability. Full details on the parareal method can be found in [42].

4.1.3 Full Approximation Scheme

The third component of PFASST is the use of the full approximation scheme (FAS). To understand the FAS, we should consider applying multigrid to solve a nonlinear equation of the form

$$A(x) = b. \tag{4.3}$$

Let $\tilde{x}$ be a provisional approximation to the solution of Eq. (4.3). The residual equation becomes

$$A(\tilde{x} + \delta) = A(\tilde{x}) + r$$

where $\delta = x - \tilde{x}$ is the error and $r = b - A(\tilde{x})$ is the residual.

Like in multigrid, we define two levels in solving Eq. (4.3): a fine level and a coarse level, denoted with the 0 and 1 superscript respectively. In multigrid, an approximation for the solution to Eq. (4.3) is first found on the fine level, $\tilde{x}^0$ by solving

$$A^0(x) = b,$$

the fine-grid formulation of Eq. (4.3). Next, one solves the residual equation on the coarse level using data from the fine-grid approximation $\tilde{x}^0$. That is, one solves

$$A^1(x) = A^1(R\tilde{x}^0)R[b - A^0(\tilde{x}^0)].$$

$A^1(x)$ and $A^0(x)$ are the formulation of the equation on the coarse and fine grid respectively; and R is a restriction operator that takes the data on the fine grid and expresses it on the coarse grid. Let $\tau^1 = A^1(R\tilde{x}^0) - RA^0(\tilde{x}^0)$. We will call τ the *FAS correction*. The residual equation can now be rewritten as

$$A^1(x) = Rb + \tau^1.$$

The FAS correction added to the residual equation allows the solution to attain fine-grid accuracy though the residual equation was calculated on the coarse gird [9]. Let the numerical approximation to solution of the residual equation be denoted as $\tilde{x}^1$. The error δ^1 is calculated by $\delta^1 = \tilde{x}^1 - R\tilde{x}^0$.

Finally, we then update the fine-grid solution $\tilde{x} = \tilde{x}^0 + L\delta^1$ where L is the interpolation operator that takes data from the coarse grid to the fine grid.

4.1.4 Parallel Full Approximation Scheme in Space and Time

The parallel full approximation scheme in space and time (PFASST) is an iterative parallel-in-time algorithm that combines SDC, parareal, and the FAS to solve ODEs and PDEs [17]. Like in multigrid methods, PFASST works by operating on multiple grids. However, PFASST operates on temporal grids with different step sizes. For now, let's assume the PFASST algorithm only has 2 levels (level 0 being the finest and level 1 the coarsest).

Like SDC, PFASST is an iterative method for solving

$$\begin{cases} y'(t) & = f(t, y(t)) \\ y(0) & = y_0 \end{cases}$$

by computing the Gaussian collocation formulation using the integral formulation for each time interval

$$\mathbf{y} - \Delta t S \mathbf{F}(\mathbf{y}) = \mathbf{y_0} \tag{4.4}$$

where $\mathbf{y} = [y(t_0), y(t_1), \cdots, y(t_M)]^T$, $\mathbf{F} = [f(t_0, y(t_0)), f(t_1, y(t_1)), \cdots, f(t_M, y(t_M))]^T$ and $\mathbf{y_0} = [y(t_0), y(t_0), \cdots, y(t_0)]^T$. It is useful to rewrite Eq. (4.4) as the following system that resembles nonlinear multigrid

$$A(\mathbf{y}) = \mathbf{b}$$

where

$$\begin{cases} A(\mathbf{y}) & = \mathbf{y} - \Delta t S \mathbf{F}(\mathbf{y}) \\ \mathbf{b} & = \mathbf{y_0}. \end{cases}$$

Similar to SDC, PFASST consists of using a provisional solution and iteratively updating that provisional solution by approximating the error via an error's equation.

In PFASST, the time domain $[0, T_N]$ is divided into N uniform intervals $[T_n, T_{n+1}]$ where each interval is assigned to a processor $\mathbf{P}_n$, $n = 0, 1, 2, ..., N-1$. When temporal and spatial parallelization are combined, $\mathbf{P}_n$ is not a single processor but one communicator in charge of a group of processors

responsible for calculating the solution within $[T_n, T_{n+1}]$ [48]. In PFASST, each interval $[T_n, T_{n+1}]$ is divided into subintervals on each level l defined by the $M_l + 1$ SDC nodes $t_l = [t_{l,0} \cdots t_{l,M}]$ such that $T_n = t_{l,0} < \cdots < t_{l,M} = T_{n+1}$. We choose that the SDC nodes to be Gauss-Lobatto nodes where there are more nodes on the fine level than the coarse level ($M_0 > M_1$).

For a two-level scheme, the system of equations that PFASST solves on each interval $[T_n, T_{n+1}]$ for $n = 0, 1, \cdots, N-1$ is

$$\mathbf{y}^0 - \Delta t S^0 \mathbf{F}^0(\mathbf{y}^0) = \mathbf{b}^0 = \mathbf{y}_0 \tag{4.5}$$

$$\mathbf{y}^1 - \Delta t S^1 \mathbf{F}^1(\mathbf{y}^1) = \mathbf{b}^1 = R\mathbf{b}^0 + \tau^1. \tag{4.6}$$

Note that the FAS correction τ^1 is given by

$$\tau^1 = A^1(R\tilde{\mathbf{y}}^0) - RA^0(\tilde{\mathbf{y}}^0).$$

Since $A(\mathbf{y}) = \mathbf{y} - \Delta t S \mathbf{F}(\mathbf{y})$, the FAS correction τ^1 becomes

$$\tau^1 = -\Delta t \left[S^1 \mathbf{F}^1(R\tilde{\mathbf{y}}^0) - RS^0 \mathbf{F}^0(\tilde{\mathbf{y}}^0) \right].$$

In the above equations, S^0 and S^1 are the spectral integration matrices corresponding to M_0 and M_1 nodes, respectively. $\mathbf{F}^0$ and $\mathbf{F}^1$ correspond to the expression of $f(t, y(t))$ on the fine and coarse grid, respectively. Like the temporal coarsening in PFASST seen with $M_0 > M_1$, PFASST allows spatial coarsening as well. That is, $\mathbf{F}^0$ may be a more accurate representation of $f(t, y)$ than $\mathbf{F}^1$. R is the restriction matrix that takes data from the fine grid to the coarse grid, and L is the integration matrix that takes data from the coarse grid to the fine grid.

PFASST starts in an initiation phase where each processor $\mathbf{P}_n$ performs n coarse SDC sweeps. This has the same computational cost as doing one SDC sweep per processor in serial; however, it has been shown that the additional SDC sweeps improve the accuracy of the final solution [17].

What follows is a description of each PFASST iterations on each processor $\mathbf{P}_n$. Assuming we have that the fine solution $\mathbf{y}_n^{[k-1]}$ of the previous iteration, the PFASST iterations undergo the following steps:

1. Do one SDC sweep on the fine level using $\mathbf{y}_n^{[k-1]}$ to calculate an approximation to the solution to Eq. (4.5). We now have an updated value $\mathbf{y}_n^{[k']}$.

2. Do n_c SDC sweeps using $\mathbf{y}_n^{[k']}$ to solve Eq. (4.6) to gain a coarse approximation $\mathbf{Y}_n^{[k]}$.

3. Interpolate the coarse correction $\tilde{\boldsymbol{\delta}}^1 = \mathbf{Y}_n^{[k]} - R\mathbf{y}_n^{[k']}$ to obtain an updated fine-grid approximation $\mathbf{y}_n^{[k]} = \mathbf{y}_n^{[k-1]} + L\tilde{\boldsymbol{\delta}}^1$.

Recall that the FAS correction in Eq. (4.6) allows PFASST to obtain fine-grid accuracy on the coarse grid. For complete details of PFASST, we recommend the reader to read [17, 48].

PFASST's parallel speedup and efficiency can be calculated by comparing PFASST to serial SDC. The speedup $\mathcal{S}$ is defined as the ratio of the cost of serial SDC to the cost of PFASST run with P_N processors in the temporal direction. The efficiency is defined as $\frac{\mathcal{S}}{P_N}$. In order to compare the two methods, we assume that PFASST and serial SDC will approximately attain the same accuracy. We will denote the number of iterations to needed to converge to the desired accuracy as K_p and K_s for the PFASST and serial SDC methods, respectively. It can be shown in [17] that one can bound the maximum speedup in the two level case by

$$\mathcal{S}(P_N) \leq \frac{K_s}{K_p} P_N.$$

The maximum parallel efficiency of PFASST is $\frac{K_s}{K_p}$, which is better than the parallel efficiency of parareal, $\frac{1}{K_p}$.

4.2 Spatial Methods

When solving PDEs, there are also considerations in the spatial calculations that the general PFASST algorithm must consider. Recall that PFASST has different temporal grids, a coarse grid and a fine grid when assuming two levels. For PDEs, PFASST attains higher efficiency by also being able to coarsen in space as well as in time.

4.2.1 Grid Systems

An example of coarsening is when the system being modeled has spatial unknowns at fixed grid points. A natural way to coarsen space in this system is by dividing the unknowns in the spatial domain by 2 in each spatial dimension, assuming d dimensions. That is, if there are $(2N)^d$ spatial unknowns on the fine level in Eq. (4.5), we can solve Eq. (4.6) using N^d spatial unknowns. This allows us to solve the coarse-grid equation with 2^d times fewer spatial unknowns. Keep in mind that the spatial coarse evaluation $\mathbf{F}^1$ in Eq. (4.6) is less accurate than the fine-grid evaluation.

4.2.2 Gridless Systems

When solving a particle system, the concept of spatial coarsening is harder to define since the spatial domain does not have a grid. Fortunately for some systems such as the N-body problem consisting of the evolution of charged particles, the fast multipole approximation Ψ allows us to an obtain spatial coarsening for a gridless system. This is done by changing the precision of Ψ by changing the amount of terms p in Ψ on the various levels. That is, we use a more accurate approximation on the fine level in Eq. (4.5) and use a less accurate approximation in Eq. (4.6).

CHAPTER 5

A PFASSTer Application: Geochemical Problem

In this chapter, we will examine how PFASST performs within a stiff system relevant to environmental science. A problem of interest for environmental scientists is the reaction and diffusion of chemical species while being advected within a fluid flow.

5.1 Formulation

We will assume that the chemical species are advected within a 1D single phase fluid flow with a known, invariant velocity. In this case, the transport of chemical species is described as

$$\frac{\partial c_i}{\partial t} = D_x \frac{\partial^2 c_i}{\partial x^2} - v_x \frac{\partial c_i}{\partial x} + R_i,\ i = 1, \ldots, n_c \tag{5.1}$$

where c_i is the concentration of species i, n_c are the number of species, t is the time, D_x is the hydrodynamic dispersion, x is the spatial dimension, v_x is the average pore velocity of the aqueous phase, and R_i represents the mass transfer and chemical reactions of c_i. Note that D_x and v_x are known inputs.

The concentrations that correspond to the species that we are solving are the following

$$
\begin{aligned}
c_1 &= [\mathrm{Ca}^{2+}] \\
c_2 &= [\mathrm{HCO}_3^-] \\
c_3 &= [\mathrm{OH}^-] \\
c_4 &= [\mathrm{H}^+] \\
c_5 &= [\mathrm{CO}_2] \\
c_6 &= [\mathrm{H}_2\mathrm{CO}_3] \\
c_7 &= [\mathrm{CO}_3^{2-}] \\
c_8 &= [\mathrm{CaHCO}_3^+]
\end{aligned}
$$

where the primary species are Ca^{2+}, HCO_3^-, and OH^-. The secondary species are H^+, CO_2, $\mathrm{H}_2\mathrm{CO}_3$, and CaHCO_3^+.

Eq. (5.1) can be simplified to the following system

$$
\begin{aligned}
\frac{\partial g_1}{\partial t} &= D_x \frac{\partial^2 g_1}{\partial x^2} - v_x \frac{\partial g_1}{\partial x} + r \\
\frac{\partial g_2}{\partial t} &= D_x \frac{\partial^2 g_2}{\partial x^2} - v_x \frac{\partial g_2}{\partial x} \\
\frac{\partial g_3}{\partial t} &= D_x \frac{\partial^2 g_3}{\partial x^2} - v_x \frac{\partial g_3}{\partial x}
\end{aligned}
\tag{5.2}
$$

where g_1, g_2, and g_3 are

$$
\begin{cases}
g_1 &= c_2 + c_5 + c_6 + c_7 + c_8, \\
g_2 &= c_2 - c_3 + c_4 + 2c_5 + 2c_6 + c_8, \\
g_3 &= c_2 + c_5 + c_6 + c_7 - c_1.
\end{cases}
\tag{5.3}
$$

r is defined by

$$
r = \kappa\sigma(1 - \Omega) \tag{5.4}
$$

where κ is a known constant; σ is a known constant, the specific reactive surface area of calcite; and Ω is a saturation index indicated whether the aqueous phase is sub-saturated or super-saturated

with respect to calcium and carbonate. Ω is defined as $\Omega = \frac{c_1 c_7}{K_{sp}}$. In our simulations, we assume $\kappa = 1.0$, $\sigma = 0.1$, and $K_{sp} = 3.98\ 10^{-9}$.

The chemical species are subject to the following constraints

$$\begin{aligned} c_6 &= (K_1 + 1)c_5 \\ c_6 &= \frac{a_2 a_4}{K_2} \\ a_2 &= \frac{a_7 a_4}{K_3} \\ a_8 &= \frac{a_1 a_2}{K_4} \\ a_3 a_4 &= K_w \end{aligned}$$

where $\mathbf{a}$ is the vector of activities, $a_i = \gamma_i c_i$, $i = 1, \ldots, 8$ and $K_1 = 1.58\ 10^{-3}$, $K_2 = 3.80\ 10^{-7}$, $K_3 = 3.72\ 10^{-11}$, $K_4 = 5.50\ 10^{-2}$, and $K_w = 4.57\ 10^{-15}$.

The activity of species i is

$$a_i = \gamma_i c_i.$$

The activity coefficient γ_i is a function of the ionic strength of the solution, which is defined as

$$\mu = \frac{1}{2} \sum_i c_i Z_i^2 \tag{5.5}$$

where Z_i is the charge of species i. The activity coefficients are computed using the DeBye-Hückel relationship

$$-\log_{10} \gamma_i = \frac{A Z_i^2 \mu^{\frac{1}{2}}}{1 + B\alpha_i \mu^{\frac{1}{2}}} \tag{5.6}$$

where A and B are values that depend upon the temperature and α_i is a coefficient for species i that depends upon the diameter of the ion solution. We set $A = 0.5$, $B = 0.326$, $\mathbf{Z} = [2\,1\,1\,1\,0\,0\,2\,1]^T$, and $\alpha = [6\,4\,3\,9\,1\,1\,5\,6]^T$. These addition algebraic constraints cause the system to be a partial differential algebraic system, which are stiff systems.

5.2 PFASST Simulation

In this dissertation, we ignore the spatial calculations in the partial differential equation system Eq. (5.2). Secondly, we add the algebraic constraints discussed earlier to obtain the following differential algebraic equation system

$$\begin{cases} \mathbf{g}_t & = \mathbf{F}(\mathbf{c}) \\ \mathbf{L}(\mathbf{c}, \mathbf{g}) & = \mathbf{0} \end{cases} \tag{5.7}$$

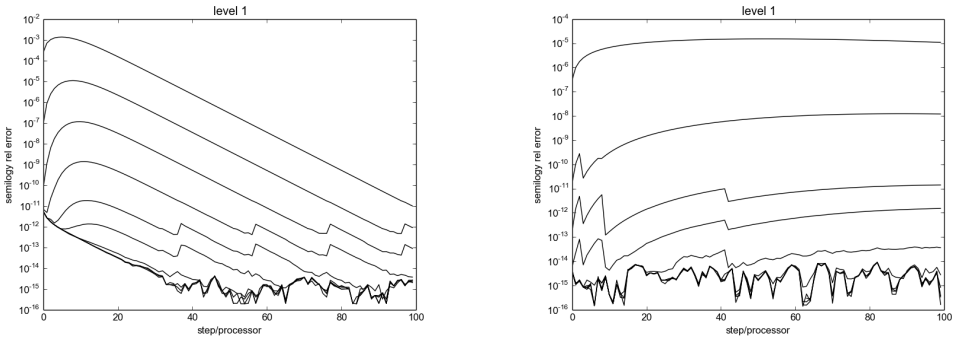
subject to the initial condition $c_3(0) = c_4(0) = 6.760177512462228\ 10^{-8}$ and $c_i(0) = 0$ when $i \neq 3,\ 4$. $\mathbf{g} = [g_i(t,x)], i = 1,2,3$; $\mathbf{F} = (r,0,0)$; $\mathbf{c} = [c_i(t,x)], i = 1,\ldots,8$; and the operator $\mathbf{L}$ is

$$\begin{cases} c_2 + c_5 + c_6 + c_7 + c_8 - g_1, \\ c_2 - c_3 + c_4 + 2c_5 + 2c_6 + c_8 - g_2 \\ c_2 + c_5 + c_6 + c_7 - c_1 - g_3 \\ c_6 - (K_1 + 1)c_5 \\ c_6 - \frac{a_2 a_4}{K_2} \\ a_2 - \frac{a_7 a_4}{K_3} \\ a_8 - \frac{a_1 a_2}{K_4} \\ a_3 a_4 - K_w. \end{cases} \tag{5.8}$$

The purpose of these simulations is to see if PFASST's parallelism can outperform the serial SDC method by converging in far less time. In the simulations, we used PFASST with 2 levels. For time stepping, we used PFASST with 100 time steps from $t \in [0,\ t_{final}]$ and varied t_{final}. The temporal coarsening proceeds as follows. Each time step $[T_n, T_{n+1}]$ used 5 and 9 Gauss-Lobatto nodes on the coarse and fine level, respectively. Since there are no spatial calculations, there is no spatial coarsening.

5.2.1 Results

What follows are the convergence results of our simulations for various end times t_{final}. We apply the serial SDC method and PFASST to the geochemical system and compare the results. The reference solution consists of serial SDC with $4,000$ time steps with 10 nodes per time step for the respective t_{final}. Finally, the reader should recall that SDC and PFASST are methods that aim to converge to the *Gauss collocation formulation* and not the exact solution. We apply serial SDC with 9 Gauss-Lobatto nodes and plot the convergence results of the relative L_∞ error of the concentrations $c_i(t)$: one with $t_{final} = 10^{-2}$ and the other with $t_{final} = 10^{-3}$.



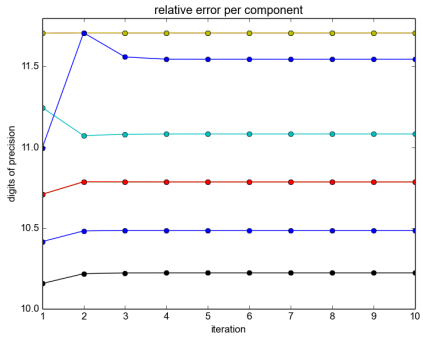
(a) $t_{final} = 10^{-2}$, $\Delta t = 10^{-4}$ (b) $t_{final} = 10^{-3}$, $\Delta t = 10^{-5}$

Figure 5.1: Relative error of $L_\infty(\mathbf{c})$ per iteration over time using SDC

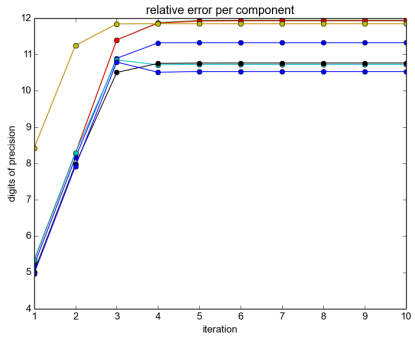
We can see that in Fig. 5.1 (a) SDC converges in 7 iterations when $t_{final} = 10^{-2}$ and 6 iterations when $t_{final} = 10^{-3}$. Notice that in Fig. 5.1 (b) the system reaches equilibrium around $t = 4\ 10^{-3}$. This occurs because $\frac{dg_1}{dt} = 0$ when $(1 - \Omega) = 0$, implying that at this point $\frac{c_1 c_7}{K_{sp}} \approx 1$. In addition to a SDC reference solution, we compare results to a Krylov deferred correction (KDC) solver. We use a converged KDC solution as a reference solution and compare the results of serial SDC. In what follows, we compare the error calculating the concentrations $c_i(t_{final})$ per iteration. Unlike SDC, these KDC methods converge to the *exact solution.*

The results in Fig. 5.2 (a) show that SDC with $t_{final} = 10^{-2}$ converges to the KDC method solution in 4 iterations. Fig. 5.2 (b) shows that SDC with $t_{final} = 10^{-3}$ converges to the KDC method solution in 5 iterations.

What follows are the convergence results of the PFASST simulations for various t_{final}. We plot the relative L_∞ error for the concentrations $c_i(t)$ between the reference solution (SDC) and



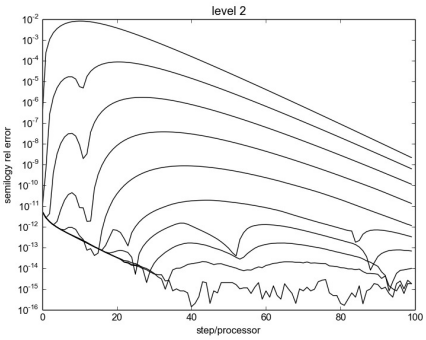
(a) $t_{final} = 10^{-2}$, $\Delta t = 10^{-4}$



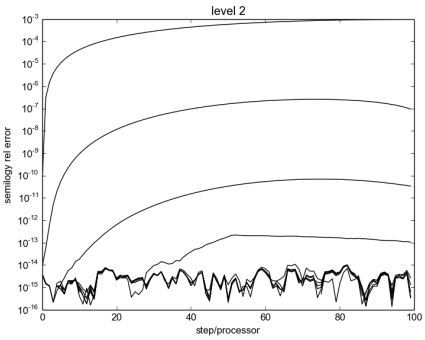
(b) $t_{final} = 10^{-3}$, $\Delta t = 10^{-5}$

Figure 5.2: Relative error per iterations for each c_i at t_{final}

approximated solution (PFASST).



(a) $t_{final} = 10^{-2}$, $\Delta t = 10^{-4}$



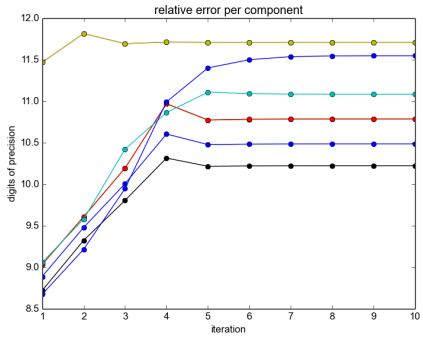
(b) $t_{final} = 10^{-3}$, $\Delta t = 10^{-5}$

Figure 5.3: Relative error of $L_\infty(\mathbf{c})$ per iteration over time, PFASST

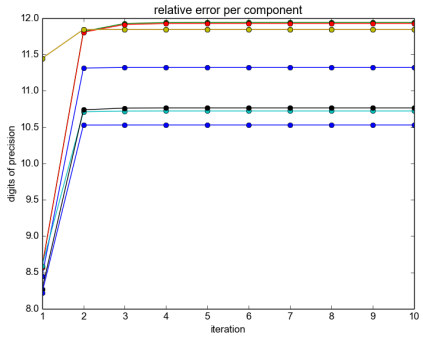
The results in Fig. 5.3 (a) show that the PFASST with $t_{final} = 10^{-2}$ converges to the reference solution in 10 iterations. When the time step is decreased to $\Delta t = 10^{-5}$, PFASST only needs 5 iterations to converge to the reference solution in Fig. 5.3 (b).

In addition to a SDC reference solution, we compare results to a Krylov deferred correction (KDC) solver. We use a converged KDC solution as a reference solution and compare the results of PFASST. In what follows, we compare the error calculating the concentrations $c_i(t_{final})$ per iteration.

The results in Fig. 5.4 (a) show that PFASST with $t_{final} = 10^{-2}$ converges to the KDC method solution in 8 iterations. Fig. 5.3 (b) shows that PFASST with $t_{final} = 10^{-3}$ converges to the KDC method solution in 4 iterations.



(a) $t_{final} = 10^{-2}$, $\Delta t = 10^{-4}$



(b) $t_{final} = 10^{-3}$, $\Delta t = 10^{-5}$

Figure 5.4: Relative error per iterations for each c_i at t_{final}

Recall for stiff systems, deferred correction methods, such as KDC, require many iterations resulting in long runtime. Thus, it is desirable to introduce parallelism in hopes of decreasing the runtime of computation. Table 5.1 shows timing results for running serial SDC and PFASST.

Table 5.1: Timing results for SDC and PFASST

Method	t_{final}	iterations	Run Time[s]	speedup	efficiency
SDC	10^{-3}	6	11.9		
PFASST	10^{-3}	5	6.5	1.83	1.83 10^{-2}
SDC	10^{-2}	7	11.4		
PFASST	10^{-2}	10	13	0.88	8.8 10^{-3}

Notice, that PFASST for this problem is not efficient compared to serial SDC. This is due to the fact that there is only temporal coarsening since we are solving an ordinary DAE system. If we added the spatial equations to solve a partial DAE, we could include spatial coarsening. This would allow an enhanced speedup and would likely make PFASST more efficient. In addition, the cost of communication between the 100 processors may be relatively high when compared to the low cost of solving for $g_i(t)$ and $c_i(t)$ in each time step, which would also decrease the efficiency.

CHAPTER 6

A PFASSTer Application: N-body Solver

Parallel integration algorithms show potential to decrease the runtime of various physical simulations such as the evolution of charged particles seen in molecular dynamics. For large systems of N particles, one of the challenges in modeling is due to the cost of direct calculation of the acceleration over all of the particles, which is $O(N^2)$. In practice, simulations avoid this difficulty by using approximation techniques such as the fast multipole method (FMM), which approximates the direct calculation with $O(N)$ [22]. Another challenge in molecular dynamics simulations lies with the fact that simulations often require large number of small time steps. Thus, parallel integration techniques such as PFASST are desirable in order to decrease the overall simulation time. We will show that parallel integration via PFASST combined with a parallel implementation of the FMM and the MRFMM (multirate fast multipole method) is capable of greatly decreasing runtime versus serial integration methods.

6.1 PFASST Simulation

In this section, we will discuss how to implement PFASST with the FMM and the MRFMM in solving Eq. (1.1). For a charged particle system with N bodies in free space, we are trying to solve the following differential equation system

$$\begin{cases} \mathbf{y}_i'(t) & = \mathbf{F}(\mathbf{y}_i(t)) \\ \mathbf{y}_i(0) & = \eta_i \;\; i = 1, \cdots, N \end{cases}$$

with

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{x}_i(t) \\ \mathbf{v}_i(t) \end{bmatrix}, \mathbf{F}(\mathbf{y}_i) = \begin{bmatrix} \mathbf{v}_i(t) \\ \mathbf{a}_i(t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_i(t) \\ -\frac{q_i}{4\pi\epsilon_0 m_i} \nabla \Psi_i(\mathbf{x}_i) \end{bmatrix}, \text{ and } \eta_i = \begin{bmatrix} \mathbf{x}_i(0) \\ \mathbf{v}_i(0) \end{bmatrix}.$$

$\mathbf{x}_i(t)$, $\mathbf{v}_i(t)$, $\mathbf{a}_i(t)$, m_i, q_i are the position, velocity, acceleration, mass, and charge of the i^{th} particle at time t. η_i is the initial condition for the i^{th} particle. For simplification, we set $4\pi\epsilon_0 m_i = 1$.

We will use the integral formulation of SDC in order to converge to the Gauss collocation formulation

$$\mathbf{y}_i - \Delta t S \mathbf{F}(\mathbf{y}_i) = \eta_i.$$

In addition, we also use the first order rectangular rule (using the left end point) $\tilde{S}$ to approximate the spectral integration matrix S.

The process of solving Eq. (2.12) at each node t_m is called an *SDC sweep*. It is useful to express the SDC sweep in matrix notation as

$$\tilde{\mathbf{y}}_i^{[k+1]} = \eta_i + \Delta t S \mathbf{F}(\tilde{\mathbf{y}}_i^{[k]}) + \Delta t \tilde{S}[\mathbf{F}(\tilde{\mathbf{y}}_i^{[k+1]}) - \mathbf{F}(\tilde{\mathbf{y}}_i^{[k]})]. \tag{6.1}$$

6.1.1 Temporal Coarsening

For all of the numerical experiments, PFASST was used with only two levels. Each time step $[T_n, T_{n+1}]$ was divided into subintervals using Gauss-Lobatto nodes. The temporal coarsening was such that there were 3 nodes per time step on the coarse level and 5 nodes per time step on the fine level.

6.1.2 Multirate Fast Multipole Method

Earlier we provided numerical results showing the different temporal scales of the FMM potential. Because the near-field forces at $\mathbf{x}_i$, $\mathbf{f}_{near} = -q_i \nabla \Psi_{near}$, change in time quickly, they are calculated at all nodes in $[T_n, T_{n+1}]$. However, the far-field forces at $\mathbf{x}_i$, $\mathbf{f}_{far} = -q_i \nabla \Psi_{far}$, change slowly in time. Instead of calculating the far-field forces at every node, we use a strategy of interpolating the far-field forces in time at some nodes and directly calculating the far-field forces at other nodes.

The following describes the interpolation scheme used on the far-field forces in the MRFMM. On the coarse level (consisting of 3 nodes), we use a constant interpolation for the far-field forces based on the value at node 0. On the fine level (consisting of five nodes), the interpolation is a degree two Lagrange interpolating polynomial based on the far-field forces from nodes 0, 2, and 4 from the previous iteration. That is, for a given particle, if we let $\mathbf{f}_i^{[k]}$ with $i = 0, 1, ..., 4$ be the

far-field forces based on iteration k at node i and l_0, l_2, and l_4 correspond to the Lagrange weights, then the far-field at nodes 1 and 3 are calculated by

$$\mathbf{f}_1^{[k]} = \mathbf{f}_0^{[k-1]} l_0(\tau_1) + \mathbf{f}_2^{[k-1]} l_2(\tau_1) + \mathbf{f}_4^{[k-1]} l_4(\tau_1) \tag{6.2}$$

$$\mathbf{f}_3^{[k]} = \mathbf{f}_0^{[k-1]} l_0(\tau_3) + \mathbf{f}_2^{[k-1]} l_2(\tau_3) + \mathbf{f}_4^{[k-1]} l_4(\tau_3) \tag{6.3}$$

where the Lagrange weights are $l_0(\tau) = \frac{(\tau-\tau_2)(\tau-\tau_4)}{(\tau_0-\tau_2)(\tau_0-\tau_4)}$, $l_2(\tau) = \frac{(\tau-\tau_0)(\tau-\tau_4)}{(\tau_2-\tau_0)(\tau_2-\tau_4)}$, and $l_4(\tau) = \frac{(\tau-\tau_0)(\tau-\tau_2)}{(\tau_4-\tau_0)(\tau_4-\tau_2)}$. τ_i corresponds to the i^{th} Lobatto node scaled in the range $[0, 1]$.

Now we will describe the calculation/ interpolation scheme for the MRFMM evaluations in PFASST. It should be said that for the FMM evaluations, we hold the FMM tree fixed within each time step $[T_n, T_{n+1}]$. The following is the MRFMM calculation/ interpolation scheme on the coarse level with 3 nodes:

Node 0 Calculate the FMM tree. Calculate $\mathbf{f}_{near}$ and $\mathbf{f}_{far}$.

Node 1 Calculate $\mathbf{f}_{near}$. Use a constant interpolation of $\mathbf{f}_{far}$ calculated at node 0.

Node 2 Calculate $\mathbf{f}_{near}$. Use a constant interpolation of $\mathbf{f}_{far}$ calculated at node 0.

The following is the MRFMM calculation/ interpolation scheme on the fine level with 5 nodes:

Node 0 Calculate the FMM tree. Calculate $\mathbf{f}_{near}$ and $\mathbf{f}_{far}$.

Node 1 Calculate $\mathbf{f}_{near}$. Interpolate $\mathbf{f}_{far}$ using Eq. (6.2).

Node 2 Calculate $\mathbf{f}_{near}$ and $\mathbf{f}_{far}$.

Node 3 Calculate $\mathbf{f}_{near}$. Interpolate $\mathbf{f}_{far}$ using Eq. (6.3).

Node 4 Calculate $\mathbf{f}_{near}$ and $\mathbf{f}_{far}$.

6.1.3 Step size

In order to have adequate multirate behavior for the near and far-field potentials, we need to choose a step size Δt such that $\Psi_{near}(\mathbf{x})$ and $\Psi_{far}(\mathbf{x})$ have adequate smoothness within the step Δt. To do this, we divide a step size $\Delta t = 1.25 \times 10^{-4}$ into 200 substeps and run the multirate test

described in section 4.2 to show the smoothness of the FMM potentials $\Psi_{near}(\mathbf{x})$ and $\Psi_{far}(\mathbf{x})$ as they change in time within the interval of size Δt.

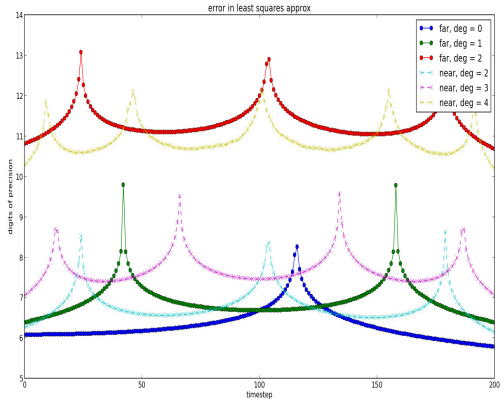


Figure 6.1: Multirate test for electrostatic case

Fig. 6.1 shows that to achieve 11 digits of precision within the interval Δt, one needs to use a least squares polynomial of degree 4 for the near-field and of degree 2 for the far-field potential. This corresponds to using 5 nodes for the far-field and 3 nodes for the near-field potential on the fine level of PFASST.

6.1.4 Spatial Coarsening

Earlier, we mentioned that for PDEs one can increase speedup by adding temporal as well as spatial coarsening to the PFASST algorithm. Because we have a particle system and there is no concept of a physical spatial grid to coarsen, we achieve spatial coarsening by changing the accuracy of the FMM evaluation on the coarse level and fine level. In Table 6.1 we show the calculation time of the FMM approximation $\Psi(x)$ over all particles for different accuracies $\epsilon = |\Phi(x) - \Psi(x)|$. Ideally

Table 6.1: Runtime for various FMM precisions

ϵ	Runtime[s]
$0.5\,10^{-2}$	$5.29\,10^{-1}$
$0.5\,10^{-6}$	$9.43\,10^{-1}$
$0.5\,10^{-9}$	1.47

we would use a computationally cheap, low-accuracy FMM approximation in the coarse propagator $\mathcal{G}$. And we would use a more computationally expensive, high-accuracy FMM approximation in

the fine propagator $\mathcal{F}$. In the numerical experiments, we surveyed the following spatial coarsening strategies:

1. Use the FMM with the same fixed spatial accuracy on the coarse and fine level.
2. Use the MRFMM with the same fixed spatial accuracy on the coarse and fine level.
3. Use the FMM with less accurate spatial solver on the coarse level and a more accurate spatial solver on the fine level.
4. Use the MRFMM with a less accurate spatial solver on the coarse level and a more accurate spatial solver on the fine level.

6.2 Numerical Results

In this section, we present the numerical results of running PFASST with the FMM and MRFMM.

6.2.1 Numerical Setup

For the following numerical experiments, we ran a simulation with 16,000 source particles with charges randomly distributed between $[-\frac{1}{2}, \frac{1}{2}]$. The particles are initially distributed within a unit cube. The simulations were run from $t = 0$ to $t = 8 \times 10^{-3}$.

The reader should know that the numerical simulations only model the Coulomb forces stemming from the potential Eq. (1.10). The simulations do not take into account the collisions of particles. Since particles of opposite charge attract, it is inevitable that at least two particles will start to accelerate towards a collision. When this occurs, we need to use a different physical model than Eq. (1.1). Therefore, we had to limit the runtime of the simulations so our physical model was valid during the total simulation time.

We have a reference solution using serial SDC with 5 nodes per time step and directly calculating the Coulomb potential Eq. (1.10). The reference solution uses 7 SDC iterations, $\Delta t = 3.125\,10^{-5}$ for 256 steps. The approximated solutions used PFASST with $\Delta t = 1.25 \times 10^{-4}$ for 64 steps. All of the PFASST calculations are done using 2 levels and 7 PFASST iterations. Each PFASST iteration consists of 1 fine SDC sweep and 1 coarse SDC sweep.

Each FMM evaluation calculates the Coulomb potential Eq. (1.10) within a certain tolerance. We will denote the tolerance approximating the Coulomb potential on the coarse and fine grid as ϵ^1 and ϵ^0, respectively. We can expect a loss of some digits of accuracy from the electrostatic potential for the acceleration due to the numerical derivatives of the gradient operator in $a_i = -q_i \nabla \Psi_i$.

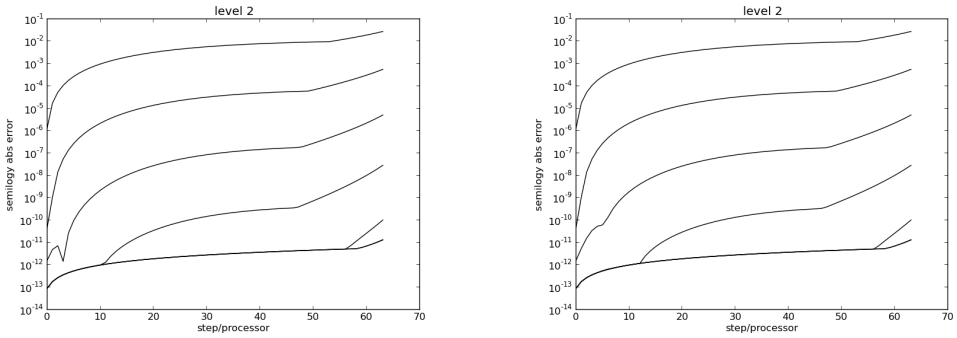
6.2.2 Results

Recall that ϵ^0 and ϵ^1 are the accuracies of the electrostatic potential evaluation on the fine and coarse level respectively. All of the PFASST solutions in this chapter have $\epsilon^0 = 0.5 \times 10^{-9}$. We will use the notation for the PFASST solutions.

- $V_{fmm0}^{[k]}$ be FMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-9}$.
- $V_{fmm1}^{[k]}$ be the FMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-6}$.
- $V_{fmm2}^{[k]}$ be the FMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-2}$.
- $V_{mr0}^{[k]}$ be the MRFMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-9}$.
- $V_{mr1}^{[k]}$ be the MRFMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-6}$.
- $V_{mr2}^{[k]}$ be the MRFMM solution on the k^{th} iteration for the velocity using a spatial solver accuracy of $\epsilon^1 = 0.5 \times 10^{-2}$.

Figures 6.2, 6.3, and 6.4 are the convergence plots of the particle velocity calculated by PFASST with various spatial solver accuracies with the FMM and MRFMM solvers. For each time step, the maximum absolute velocity error of the approximated solution compared to the reference solution over all particles is plotted for each iteration. Each line corresponds to the absolute error at each time step of a PFASST iteration with the topmost line corresponding to the first iteration. The next line down corresponds to the second iteration and so on.

In Fig. 6.2, both solutions used a spatial solver with electrostatic potential tolerance $\epsilon^0 = \epsilon^1 = 0.5 \times 10^{-9}$. We see that as we increase the PFASST iteration, the errors decrease. Here we see that the MRFMM scheme has similar convergence properties as the FMM scheme. The numerical method converges in 6 iterations. However we will later that PFASST with the MRFMM runs quicker than PFASST with the FMM.



(a) convergence of $V_{fmm0}^{[k]}$ (b) convergence of $V_{mr0}^{[k]}$

Figure 6.2: Absolute error of velocity per iteration compared to the reference solution. Coarse-level FMM precision is $\epsilon^1 = 0.5 \times 10^{-9}$

Instead of fixing the spatial accuracy on the coarse and fine level, we can also use different spatial accuracies on the various levels. Notably, we can do the following.

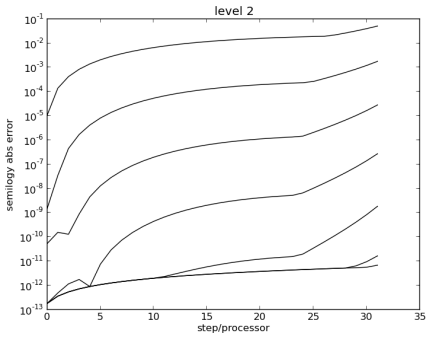
1. Use a less accurate FMM on the coarse level and a more accurate FMM on the fine level.

2. Use a less accurate MRFMM on the coarse level and a more MRFMM on the fine level.

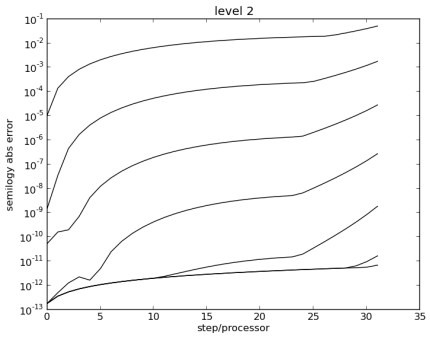
Figures 6.3 and 6.4 show the velocity convergence plots of the FMM and MRFMM by varying the spatial accuracy on the coarse level and fixing the spatial accuracy on the fine level. It seems that changing the spatial accuracy of the correction equation has little effect on the convergence behavior of the solution.

In order to see the effect of varying the spatial accuracy on the correction equation, we calculate the L_2 norm of the relative error of the various FMM/ MRFMM solutions ($V_{mr0}^{[k]}$, $V_{mr1}^{[k]}$, $V_{fmm1}^{[k]}$, $V_{fmm2}^{[k]}$) with respect to the high accuracy FMM solution, $V_{fmm0}^{[k]}$ of that particular iteration k for all iterations. We show the results in Figures 6.5, 6.6, and 6.7.

Figure 6.6 shows that the FMM solution $V_{fmm1}^{[k]}$ converges to the more accurate solution $V_{fmm0}^{[k]}$ within 3 iterations and $V_{fmm2}^{[k]}$ does so in 4 in Fig. 6.5. Since we showed earlier that it takes 6 PFASST

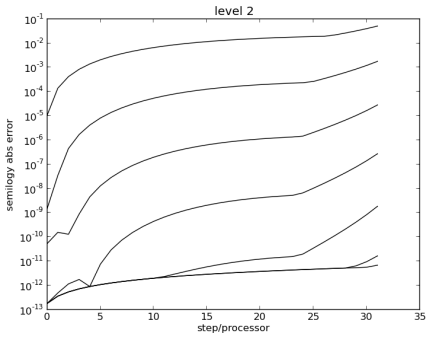


(a) convergence of $V_{fmm1}^{[k]}$

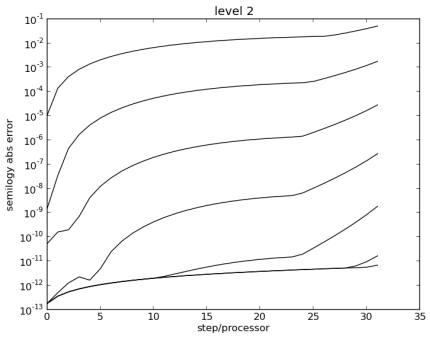


(b) convergence of $V_{mr1}^{[k]}$

Figure 6.3: Absolute error of velocity per iteration compared to the reference solution. Coarse-level FMM precision is $\epsilon^1 = 0.5 \times 10^{-6}$

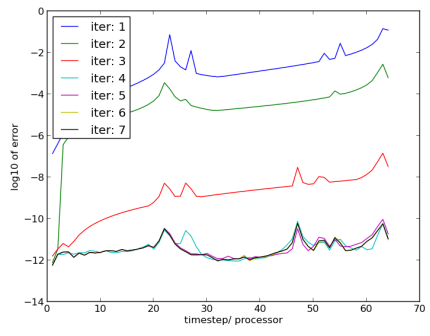


(a) convergence of $V_{fmm2}^{[k]}$

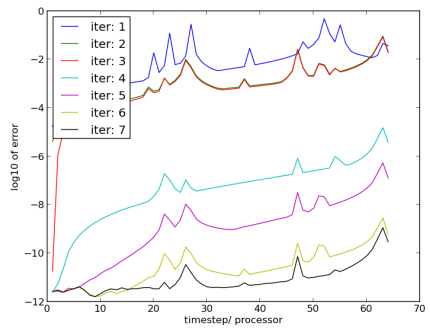


(b) convergence of $V_{mr2}^{[k]}$

Figure 6.4: Absolute error of velocity per iteration compared to the reference solution. Coarse-level FMM precision is $\epsilon^1 = 0.5 \times 10^{-2}$

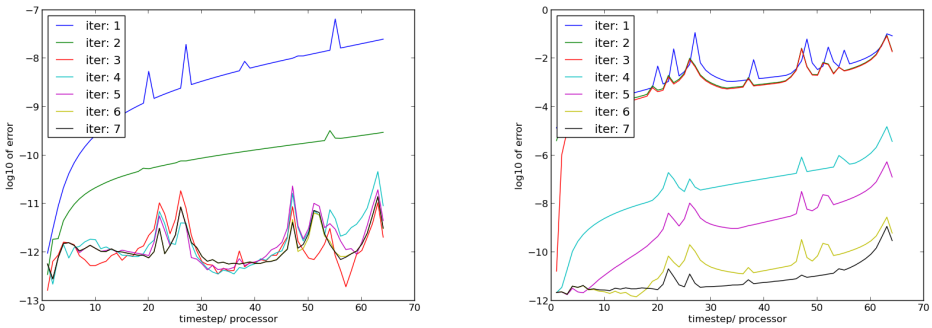


(a) relative error of $V_{fmm2}^{[k]}$



(b) relative error of $V_{mr2}^{[k]}$

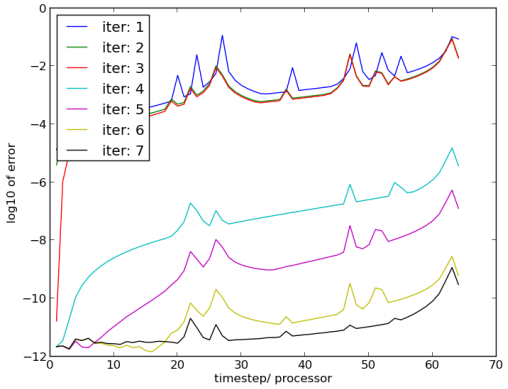
Figure 6.5: Relative error of velocity per iteration to $V_{fmm0}^{[k]}$ with coarse-level FMM precision $\epsilon^1 = 0.5 \times 10^{-2}$



(a) relative error of $V^{[k]}_{fmm1}$

(b) relative error of $V^{[k]}_{mr1}$

Figure 6.6: Relative error of velocity per iteration to $V^{[k]}_{fmm0}$ with coarse-level FMM precision $\epsilon^1 = 0.5 \times 10^{-6}$



(a) relative error of $V^{[k]}_{mr0}$

Figure 6.7: Relative error of velocity per iteration to $V^{[k]}_{fmm0}$ with coarse-level FMM precision $\epsilon^1 = 0.5 \times 10^{-9}$

iterations for the FMM solution to converge, dropping the accuracy of each FMM evaluation in solving Eq. (4.6) on the coarse level has negligible effect in the overall convergence.

The MRFMM solutions $V_{mr0}^{[k]}$, $V_{mr1}^{[k]}$, $V_{mr2}^{[k]}$ are less accurate than $V_{fmm0}^{[k]}$ and converge towards the $V_{fmm0}^{[k]}$ solution slower than $V_{fmm1}^{[k]}$. The difference in behavior from both MRFMM solutions compared to the FMM solutions comes from the MRFMM's interpolation of the far-field. Noticeably, we lose some far-field accuracy because of the constant interpolation of the far-field per time step when solving Eq. (4.6).

6.2.3 The Residual Equation

In order to explain the lack of change in the convergence behavior of PFASST using the FMM and MRFMM, while varying the spatial accuracy, we need to examine the coarse-level residual equation Eq. (4.6). We can express the residual equation as

$$\mathbf{y}^1 - \Delta t S^1 \mathbf{F}^1(\mathbf{y}^1) \ = \mathbf{y}_0^1 - \Delta t[\, S^1 \mathbf{F}^1(R\tilde{\mathbf{y}}^0) - RS^0 \mathbf{F}^0(\tilde{\mathbf{y}}^0)\,]. \tag{6.4}$$

We solve the above equation by using one SDC sweep to obtain

$$\tilde{\mathbf{y}}^1 = \mathbf{y}_0^1 + \Delta t RS^0 \mathbf{F}^0(\tilde{\mathbf{y}}^0) + \Delta t \tilde{S}^1[\mathbf{F}^1(\tilde{\mathbf{y}}^1) - \mathbf{F}^1(R\tilde{\mathbf{y}}^0)]. \tag{6.5}$$

For this analysis, we are only focussing on solving for the velocity as the unknown $\mathbf{y}$ as opposed to having $\mathbf{y} = [\mathbf{x}\,\mathbf{v}]^T$. Hence, $\mathbf{F}$ will be the acceleration calculation. We can rewrite $\mathbf{F}(\mathbf{y})$ as the acceleration due to the sum of near and far-field forces such that $\mathbf{F}(\mathbf{y}) = \mathbf{F}_{near}(\mathbf{y}) + \mathbf{F}_{far}(\mathbf{y})$. Eq. (6.5) becomes

$$\begin{aligned} \tilde{\mathbf{y}}^1 = \tilde{\mathbf{y}}_0 &+ \Delta t RS^0 \mathbf{F}^0(\tilde{\mathbf{y}}^0) \\ &+ \Delta t \tilde{S}^1[\mathbf{F}_{near}^1(\tilde{\mathbf{y}}^1) - \mathbf{F}_{near}^1(R\tilde{\mathbf{y}}^0)] + \Delta t \tilde{S}^1[\mathbf{F}_{far}^1(\tilde{\mathbf{y}}^1) - \mathbf{F}_{far}^1(R\tilde{\mathbf{y}}^0)]. \end{aligned} \tag{6.6}$$

Let's examine the case of using the FMM with varying spatial accuracy on the coarse level. Recall that V_{fmm0}, V_{fmm1}, and V_{fmm2} have the spatial accuracy of the potential set to $\epsilon^0 = 0.5 \times 10^{-9}$ on the fine level. First note that by using the FAS, the coarse-level solution is able to attain fine-grid

resolution via the

$$\tilde{\mathbf{y}}^0 + \Delta t R S^0 \mathbf{F}^0(\tilde{\mathbf{y}}^0)$$

term in Eq. (6.6). Given the same fine-level solution $\tilde{\mathbf{y}}^0$, for all of the FMM solutions, this operation is the same. The difference in the solutions comes from the later part of Eq. (6.6), which is

$$\Delta t \tilde{S}^1[\mathbf{F}^1_{near}(\tilde{\mathbf{y}}^1) - \mathbf{F}^1_{near}(R\tilde{\mathbf{y}}^0)] + \Delta t \tilde{S}^1[\mathbf{F}^1_{far}(\tilde{\mathbf{y}}^1) - \mathbf{F}^1_{far}(R\tilde{\mathbf{y}}^0)].$$

By examining the solutions V_{fmm0}, V_{fmm1}, and V_{fmm2}, we see that the difference in solutions comes from the coarse-level calculation of near and far-field forces in Eq. (6.6). To see that relation, let $\mathbf{x}$ be $R\tilde{\mathbf{y}}^0$ and $\mathbf{u}$ be the solution to the SDC sweep $\tilde{\mathbf{y}}^1$ in Eq. (6.5) such that $\mathbf{u} = \mathbf{x} + \delta$. We can express $\mathbf{F}^1(\mathbf{u}) - \mathbf{F}^1(\mathbf{x})$ for the i^{th} particle in Eq. (6.5) as

$$\begin{aligned}
F^1(u_i) - F^1(x_i) &= -q_i \nabla [\Phi(u_i) - \Phi(x_i)] \\
&= -q_i \nabla \sum_j q_j \left(\frac{1}{|u_j - u_i|} - \frac{1}{|x_j - x_i|} \right) \\
&= -q_i \nabla \sum_j q_j \left(\frac{1}{|x_j - x_i + \delta_j - \delta_i|} - \frac{1}{|x_j - x_i|} \right) \\
&= -q_i \nabla \sum_j \frac{q_j}{|x_j - x_i|} \left(\left(1 + \frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right)^{-1} - 1 \right).
\end{aligned}$$

Recall we have set $4\pi\epsilon_0 = 1$. Taylor expanding the above formulation and assuming $\frac{|\delta_j - \delta_i|}{|x_j - x_i|} < 1$, we obtain

$$\begin{aligned}
F^1(u_i) - F^1(x_i) &= -q_i \nabla \sum_j \frac{q_j}{|x_j - x_i|} \sum_{k=1} (-1)^k \left(\frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right)^k \\
F^1(u_i) - F^1(x_i) &= q_i \nabla \left[\sum_j \frac{q_j}{|x_j - x_i|} \left(\frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right) + O\left(\left(\frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right)^2 \right) \right]. \qquad (6.7)
\end{aligned}$$

This can be rewritten as

$$F^1(u_i) - F^1(x_i) \approx q_i \nabla \left[\sum_{x_j \in \Omega_{near}} \frac{q_j}{|x_j - x_i|} \left(\frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right) + \sum_{x_j \in \Omega_{far}} \frac{q_j}{|x_j - x_i|} \left(\frac{|\delta_j - \delta_i|}{|x_j - x_i|} \right) \right]. \qquad (6.8)$$

Recall that each FMM calculation directly calculates near-field forces and approximates far-field forces. The more accurate the FMM approximation, the more particles are assumed in the near-field. The less accurate the FMM, the smaller the near-field is, and the larger the far-field is. Hence, the inaccuracy from the FMM comes from not only limiting the amount of terms in the expansion but also poorly approximating the far-field by not including enough near-field particles in direct calculation. However, even for an inaccurate FMM calculation, the potential for the nearest (hence, the most influential), neighbors to a particle are always calculated directly.

By using less accurate FMM evaluation on the coarse level than $V_{fmm0}^{[k]}$, $V_{fmm1}^{[k]}$ and $V_{fmm2}^{[k]}$ are able to attain similar convergence to $V_{fmm0}^{[k]}$ after few iterations as seen in Figures 6.6 and 6.5 with a smaller computation cost. This occurs because the difference in the far-field forces in Eq. (6.8) are negligible. Note that as approximations become more accurate as we iterate, the corrections in $|\delta_j - \delta_i|$ become smaller. More importantly, in the far-field $|x_j - x_i|$ is large. This dampens the effect of the calculated inaccuracies even when the FMM solver is inaccurate.

The difference per iteration between $V_{fmm0}^{[k]}$, $V_{fmm1}^{[k]}$, and $V_{fmm2}^{[k]}$ comes from the near-field component in Eq. (6.8). We see that even though $V_{fmm1}^{[k]}$ and $V_{fmm2}^{[k]}$ contain an inaccurate spatial solver on the coarse level, the two less accurate solutions directly calculate enough of the most influential near-field particles. Hence, they $V_{fmm1}^{[k]}$ and $V_{fmm2}^{[k]}$ are able to obtain similar convergence behavior to $V_{fmm0}^{[k]}$ in few iterations.

There is very similar convergence behavior with $V_{mr0}^{[k]}$, $V_{mr1}^{[k]}$, and $V_{mr2}^{[k]}$ in Figures 6.7, 6.6, and 6.5. However, the MRFMM solutions are not as able to converge to the $V_{fmm0}^{[k]}$ solution as the other FMM solutions. This difference in behavior comes from the MRFMM's *constant* interpolation of the far-field forces over a time step Δt on the coarse level in Eq. (6.8). As seen in Figure 6.1, using a constant interpolation for the far-field forces causes a loss in accuracy. But a second order interpolation of the far-field (used on the fine level) does not. Hence, unlike the FMM solutions, the dominant error in Eq. (6.8) comes from the far-field interpolation and not the near-field calculations for the MRFMM solutions. Recall that the near-field forces are calculated at every node in a time step Δt in both the MRFMM and the FMM. Even with the interpolated far-field forces, the MRFMM solutions are able to converge very close to the high spatial accuracy FMM solution $V_{fmm0}^{[k]}$ with less computational cost after 6 iterations.

6.2.4 Gravitational Forces

Each particle in the electrostatic simulation is under the influence of more singular attractive forces ($\lim_{r\to 0} \frac{1}{r^2}$) and smoother repulsive forces ($\lim_{r\to \infty} \frac{1}{r^2}$) due to charge-polarities being opposing or similar, respectively. If all the forces were to be attractive, like in gravitational simulations, we should expect the MRFMM to be less effective. This should occur because the force field exclusively contains increasingly fast-changing attractive forces that are less smooth in time. Thus, these forces are more difficult to interpolate.

To show this phenomena, we run the same simulations as earlier with the following changes. All of the charges have the same magnitude and are all positive. We change the sign in Eq. (1.1) so that now all of forces are attractive whereas they would be repulsive. Finally, the time step is shortened to $\Delta t = 7.8125 \times 10^{-5}$ to avoid the more imminent particle collisions.

Figure 6.8 shows the result of the multirate test within the first time step Δt. Keep in mind that the forces are less singular in the first time step. Figures 6.9 and 6.10 are the velocity convergence

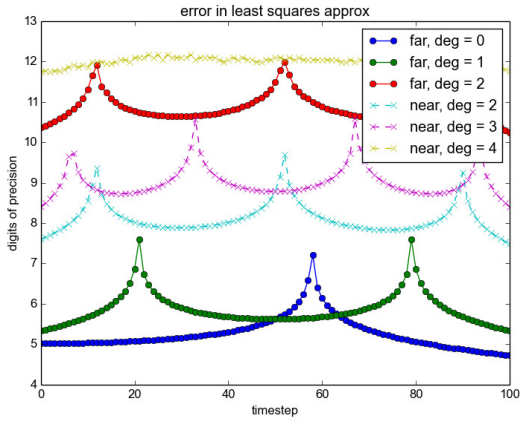
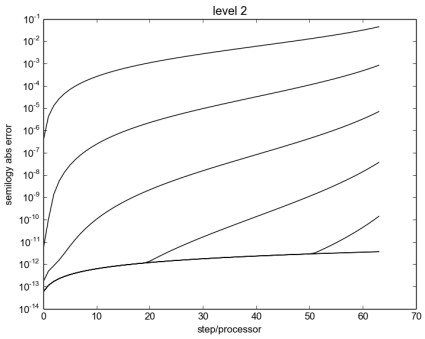
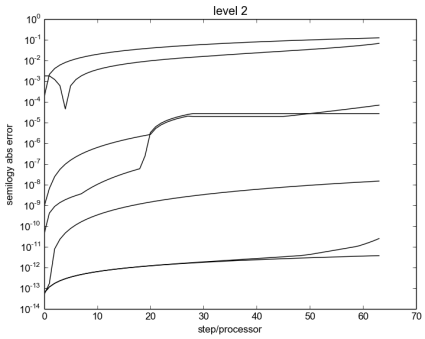


Figure 6.8: Multirate test for gravitational case

plots for PFASST with the FMM and MRFMM respectively. The results show that the MRFMM converges to the solution; however, it takes more iterations than in the FMM case. In addition, the initial MRFMM iterations are less accurate than the FMM's. The higher abundance of singular forces causes the MRFMM to behave less predictively than in the simulations with the dual charge polarities.

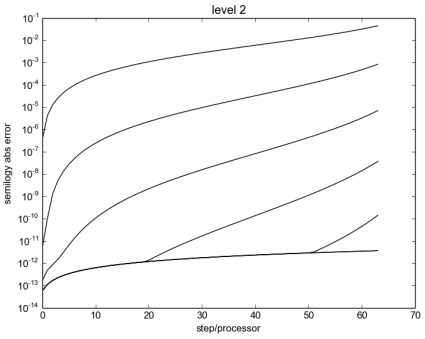


(a) convergence of $V_{fmm0}^{[k]}$

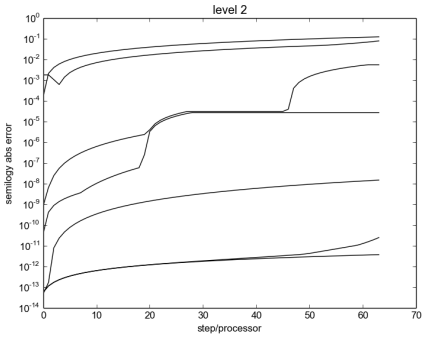


(b) convergence of $V_{mr0}^{[k]}$

Figure 6.9: Absolute error of velocity per iteration compared to the reference solution. Coarse-level FMM precision is $\epsilon^1 = 0.5 \times 10^{-9}$



(a) convergence of $V_{fmm2}^{[k]}$



(b) convergence of $V_{mr2}^{[k]}$

Figure 6.10: Absolute error of velocity per iteration compared to the reference solution. Coarse-level FMM precision is $\epsilon^1 = 0.5 \times 10^{-2}$

6.3 Speedup

The following are results for the parallel speedup and efficiency for the PFASST algorithm using the FMM and MRFMM for the electrostatic problem. Recall that ϵ^0 and ϵ^1 correspond to the FMM evaluation tolerance for the electrostatic potential for the fine and coarse level, respectively.

Table 6.2: Serial SDC, 6 iterations

Method	ϵ^0	n_{steps}	Run Time[s]
FMM	$0.5\,10^{-9}$	64	4131
MRFMM	$0.5\,10^{-9}$	64	3275

Table 6.3: PFASST, 7 iterations

Method	ϵ^1	ϵ^0	n_{steps}	Speedup	Parallel Efficiency	Run Time[s]
FMM	$0.5\,10^{-2}$	$0.5\,10^{-9}$	64	15.078	0.236	274
FMM	$0.5\,10^{-6}$	$0.5\,10^{-9}$	64	8.101	0.127	510
FMM	$0.5\,10^{-9}$	$0.5\,10^{-9}$	64	5.819	0.091	710

Table 6.4: PFASST, 7 iterations

Method	ϵ^1	ϵ^0	n_{steps}	Speedup	Parallel Efficiency	Run Time[s]
MRFMM	$0.5\,10^{-2}$	$0.5\,10^{-9}$	64	26.484	0.414	156
MRFMM	$0.5\,10^{-6}$	$0.5\,10^{-9}$	64	9.675	0.151	427
MRFMM	$0.5\,10^{-9}$	$0.5\,10^{-9}$	64	7.915	0.124	522

Table 6.2 shows the runtime of serial SDC. Tables 6.3 and 6.4 shows the various speedup attained by PFASST. PFASST is able to run much quicker than serial SDC. The various speedup in the PFASST solutions comes from mainly two different attributes: the spatial coarsening and the multirate time stepping. By making the coarse-level spatial solver less accurate, i.e. cheaper to compute, PFASST is able to undergo the initialization phase described in [17] much quicker. This allows PFASST to start iteratively approximating the solutions much earlier than if we used a high accuracy, i.e.more computationally expensive, spatial solver on the coarse level. This phenomena is greatly enhanced when the MRFMM is used because the multirate time stepping has fewer calculations. The PFASST iterations are also done quicker due to the coarser approximations and MRFMM.

To explain the speedup that PFASST has with the MRFMM, let $\mathcal{F}$ and $\mathcal{N}$ be the amount of work it takes for a FMM evaluation to calculate the far-field forces and near-field respectively. For

this analysis, we will assume that that the coarse and fine-level spatial solver will have the same accuracy. We will ignore the cost of communication between processors and the cost of interpolation and restriction between PFASST levels. For the FMM, computing 1 PFASST iteration (with 1 SDC sweep on the fine and coarse level) corresponds to doing $2 \times 3(\mathcal{F} + \mathcal{N})$ work on the coarse level (3 nodes) and $2 \times 5(\mathcal{F} + \mathcal{N})$ work on the fine level (5 nodes) for a total of amount of work $\mathcal{T}_{fmm} = 16\mathcal{F} + 16\mathcal{N}$ amount of work. For the MRFMM, computing 1 PFASST iteration corresponds to doing $2 \times (1\mathcal{F} + 3\mathcal{N})$ work on the coarse level and $2 \times (3\mathcal{F} + 5\mathcal{N})$ work on the fine level for a total of amount of work $\mathcal{T}_{mr} = 8\mathcal{F} + 16\mathcal{N}$.

Let $\beta = \frac{\mathcal{T}_{fmm}}{\mathcal{T}_{mr}}$ be the ratio of the amount of work between PFASST with the FMM and the MRFMM. We have two limiting cases for β: when the amount of work in the far field equals the amount of work done in the near field ($\mathcal{F} = \mathcal{N}$) and when the amount of work in the far field dominates that of the near field ($\mathcal{F} \gg \mathcal{N}$). This gives us bounds for β as long as $\mathcal{F} \geq \mathcal{N}$

$$\frac{4}{3} \leq \beta \leq 2\frac{1 + \frac{\mathcal{N}}{\mathcal{F}}}{1 + 2\frac{\mathcal{N}}{\mathcal{F}}}. \tag{6.9}$$

Eq. (6.9) provides insight on how the MRFMM algorithm should be made to increase speed in PFASST. To increase the speedup, we need $\frac{\mathcal{N}}{\mathcal{F}} \ll 1$. The smaller this ratio is, the faster the MRFMM will run in comparison to the regular FMM. That is, one should design a FMM algorithm in a way such that as much of the computational time is used calculating the far-field as possible.

6.4 Conclusion

For the simulation of charged particles, we have showed that parallel integration in time should be considered. The PFASST algorithm used with the FMM allows one to attain speedup versus serial SDC. PFASST's multigrid-like structure allows additional speedup by relaxing the accuracy of the FMM on the coarse level without compromising the convergence of the solution. By taking into account the temporal behavior of the far-field and near-field forces, one can use the MRFMM's property of interpolating the far-field forces to decrease the total amount of calculations per time step. Hence, PFASST used with the MRFMM allows for greater speedup without too much of a loss of accuracy.

The MRFMM decreases the amount of calculations per time step by only calculating forces when they change quickly in time and interpolating forces when the change slowly in time. This is a promising property for more complex simulations that include dipole ($\frac{1}{r^3}$) and Van der Waals ($\frac{1}{r^7}$) forces. These more singular forces have the property that the more singular the force, the smaller Ω_{near} is and the larger Ω_{far} is. Hence, by using the MRFMM, one can decrease the net amount of calculation by interpolating the larger far-field in a time step Δt instead of calculating the far-field at each node within the time step.

CHAPTER 7

Future Work

When using deferred correction methods, a numerical algorithm of an "optimal" method can be made by keeping in mind the ideas of the "collocation formulation" and "convergence procedure." For a given problem, an efficient method can made by selecting a collocation formulation to converge to based on the properties of the solution from a "collocation formulation database." Afterwards, different deferred corrections schemes can be selected from the "deferred correction methods database" to effectively reduce different error components in the provisional solution. When deferred correction methods stall, especially in stiff systems, the modified JFNK then can be used to accelerate convergence. A possible way to make this new JFNK method more efficient would be to find a way to reuse/ recycle the old Krylov basis vectors instead of always computing new Krylov basis in forming a new subspace.

The PFASST algorithm has great potential in decreasing runtime by taking into account temporal parallelism. PFASST could be used to accelerate calculations of many elliptic equation systems such as Stokes flows, electromagnetics, and Helmholtz equation systems. For elliptic equation solvers and fast N-body solvers, work can be done to increase efficiency by using temporal multirate techniques when taking account more singular potentials than the $\frac{1}{r}$ Coulomb potential. Multirate time stepping with these singular potentials would greatly add to the algorithm's efficiency.

For differential (algebraic) equation initial value problems or very stiff systems, PFASST should be able to be combined with the modified JFNK. Since PFASST depends on SDC as its integrator, PFASST will suffer from order reduction or divergence in solving extremely stiff systems. However, the inclusion of the modified JFNK could overcome this limitation while still maintaining time parallelism.

PFASST's use of multiple grids and multirate time stepping are a significant step towards possible temporally adaptive algorithms where signals are only calculated when they change significantly and interpolated when they do not. This feature added with time-parallelism can be very promising

for long-time, large-scale simulations.

APPENDIX A

PROOFS OF THEOREMS

A.1 Proof of Theorem 2.2.3.

Proof. Assuming p points $\{1/p, 2/p, \cdots, (p-1)/p, 1\}$ are used in the uniform collocation formulation, then $\tilde{S}$ is a lower triangular matrix and all non-zero entries (including diagonal entries) are $1/p$. Simple calculation shows that $\tilde{S}^{-1}$ has zero entries everywhere except along the diagonal and subdiagonal, with nonzero entries p on the diagonal and $-p$ on the subdiagonal,

$$\tilde{S}^{-1} = \begin{bmatrix} p & 0 & 0 & \cdots & 0 & 0 \\ -p & p & 0 & \cdots & 0 & 0 \\ 0 & -p & p & \cdots & 0 & 0 \\ 0 & 0 & -p & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -p & p \end{bmatrix}.$$

Consider the vector $\mathbf{V}_j = [(p-1)^j, (p-2)^j, \cdots, 2^j, 1^j, 0]^T$ $(j = 1, \cdots, p-1)$ and $\mathbf{V}_0 = [1, 1, \cdots, 1, 1]^T$. As S integrates polynomials of degree $\leq p-1$ exactly, one can show

$$(\tilde{S}^{-1}S - I)\mathbf{V}_j = \frac{1}{j+1}\sum_{l=0}^{j-1}\binom{l}{j+1}\mathbf{V}_l,$$

and

$$(\tilde{S}^{-1}S - I)\mathbf{V}_0 = \mathbf{0}.$$

Define $\mathbf{W}_0 = \mathbf{V}_0$. The basis for the Jordan canonical form can then be constructed recursively by solving $(\tilde{S}^{-1}S - I)\mathbf{W}_j = \mathbf{W}_{j-1}$, where $\mathbf{W}_j$ consists of a linear combination of $\mathbf{V}_k$, $k = 0, \cdots, j$. □

A.2 Proof of Theorem 2.2.4.

We start from the following Lemma:

Lemma A.2.1. *For the trapezoidal rule preconditioned uniform collocation formulation (InDC-*

yp-T), the matrix $S - \tilde{S}$ maps the vector $[(\frac{j}{p})^k]_{j=0}^p := [(\frac{0}{p})^k, (\frac{1}{p})^k, (\frac{2}{p})^k, \cdots, (\frac{p-1}{p})^k, 1]^T$ ($k \leq p$) to a linear combination of vectors $[(\frac{j}{p})^m]_{j=0}^p$, $m = 0, \cdots, k-1$.

Proof. Assume $p+1$ points $\{0/p, 1/p, 2/p, \cdots, (p-1)/p, 1\}$ are used in the uniform collocation formulation. As the integration matrix S integrates polynomials of degree p or less exactly, we have

$$S\left[(\frac{j}{p})^k\right]_{j=0}^p = \left[\int_0^{\frac{j}{p}} x^k dx\right]_{j=0}^p = \frac{1}{k+1}\left[(\frac{j}{p})^{k+1}\right]_{j=0}^p.$$

Now consider the j^{th} entry of the vector $\tilde{S}[(\frac{j}{p})^k]_{j=0}^p$ given by

$$\begin{aligned}
\tilde{S}\left[(\frac{j}{p})^k\right]_j &= \frac{1}{p}\left(\frac{1}{2}\left(\frac{0}{p}\right)^k + \left(\frac{1}{p}\right)^k + \cdots + \left(\frac{j-1}{p}\right)^k + \frac{1}{2}\left(\frac{j}{p}\right)^k\right) \\
&= \frac{1}{p^{k+1}}(1^k + 2^k + \cdots + j^k - \frac{1}{2}j^k) \\
&= \frac{1}{p^{k+1}}(\frac{j^{k+1}}{k+1} + \frac{1}{2}j^k + \text{ lower order } (<k) \text{ terms } - \frac{1}{2}j^k).
\end{aligned}$$

Therefore, after cancelling the j^{k+1} and j^k terms, we have

$$(S - \tilde{S})\left[(\frac{j}{p})^k\right]_{j=0}^p = \sum_{m=0}^{k-1} c_m \left[(\frac{j}{p})^m\right]_{j=0}^p.$$

□

Applying Lemma A.2.1 and the Taylor expansion of the initial provisional solution in the trapezoidal rule preconditioned deferred correction iterations for the uniform collocation formulation (InDC-yp-T), Theorem 2.2.4 can be proved as follows:

Proof. From Eq. (2.34), we see that the correction matrix has the expansion

$$C_{ns}^t = (\lambda\Delta t)(\mathbf{S} - \tilde{\mathbf{S}}) + (\lambda\Delta t)^2\tilde{\mathbf{S}}(\mathbf{S} - \tilde{\mathbf{S}}) + (\lambda\Delta t)^3\tilde{\mathbf{S}}^2(\mathbf{S} - \tilde{\mathbf{S}}) + \cdots,$$

and the initial provisional solution b has the expansion of the form (neglecting all $(\Delta t)^{p+1}$ and higher order terms)

$$b \approx \sum_{m=0}^{p} (\lambda\Delta t)^m c_m \left[(\frac{j}{p})^m\right].$$

By induction and Lemma A.2.1, it is straightforward to show that

$$(C_{ns}^t)^k b \approx (\lambda \Delta t)^{2k} \sum_{m=0}^{p} c_{m,k} \left[(\frac{j}{p})^m \right],$$

neglecting $(\Delta t)^{p+1}$ and higher order terms. Therefore, after each trapezoidal rule preconditioned SDC iteration for the uniform collocation formation, the order will increase by $(\Delta t)^2$, until it reaches $(\Delta t)^{p+1}$. □

REFERENCES

[1] U.M. Ascher and L.R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, SIAM, 1998.

[2] K. Atkinson, *An introduction to numerical analysis, 2nd edition*, John Wiley, 1989.

[3] W. Auzinger, H. Hofstatter, W. Kreuzer, and E. Weinmuller, *Modified defect correction algorithms for ODEs. part i: General theory*, Numerical Algorithms **36** (2004), 135–156.

[4] R. Barrett et al., *Templates for the solution of linear systems: Building blocks for iterative methods, 2nd edition*, SIAM, Philadelphia, 1994.

[5] R. Barrio, *On the A-stability of Runge-Kutta collocation methods based on orthogonal polynomials*, SIAM Journal Numerical Analysis **36** (1999), no. 4, 1291–1303.

[6] G. Beylkin and K. Sandberg, *ODE solvers using band-limited approximations*, Journal of Computational Physics **265** (2014), 156–171.

[7] A. Bourlioux, A.T. Layton, and M. Minion, *High-order multi-implicit spectral deferred correction methods for problems of reactive flow*, Journal of Computational Physics **189** (2003), 351–376.

[8] E.L. Bouzarth and M. Minion, *A multirate time integrator for regularized Stokeslets*, Journal of Computational Physics **229** (2010), no. 11, 4208–4224.

[9] W. Brigss, V. Henson, and S. McCormick, *A multigrid tutorial*, SIAM, 2000.

[10] P.N. Brown, A.C. Hindmarsh, and L.R. Petzold, *Using Krylov methods in the solution of large-scale differential-algebraic systems*, SIAM Journal of Scientific Computing **15** (1994), 1467–1488.

[11] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang, *Spectral methods in fluid dynamics*, Springer-Verlag, 1988.

[12] M. Causley, A. Christlieb, B. Ong, and L. Van Groningen, *Method of lines transpose: An implicit solution to the wave equation*, Mathematics of Computation **83** (2014), 2763–2786.

[13] W. Chen, X. Wang, and Y. Yu, *Reducing the computational requirements of the differential quadrature method*, Numerical Methods for Partial Differential Equations **12** (1996), 565–577.

[14] A. Christlieb, B. Ong, and J. Qiu, *Integral deferred corrections constructed with high order Runge-Kutta methods*, Mathematics of Computation **79** (2010), 761–783.

[15] X. Du and D.B. Szyld, *Inexact GMRES for singular linear systems*, 2007.

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

[17] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Communications in Applied Mathematics and Computational Science **7** (2012), 105–132.

[18] M.J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM Journal on Scientific Computing **29** (2007), 556–578.

[19] A. Glaser and V. Rokhlin, *A new class of highly accurate solvers for ordinary differential equations*, Journal of Scientific Computing **38** (2009), no. 3, 368–399.

[20] D. Gottlieb and S.S. Orszag, *Numerical analysis of spectral methods*, SIAM, Philadelphia, 1977.

[21] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM Journal of Numerical Analysis **28** (1991), 1071–1080.

[22] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics **73** (1987), 325–348.

[23] E. Hairer and M. Hairer, *Gnicodes - matlab programs for geometric numerical integration*, Frontiers in Numerical Analysis, Springer, Berlin, 2003, pp. 199–240.

[24] E. Hairer, C. Lubich, and M. Roche, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, Springer-Verlag, 1989.

[25] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations*, Springer-Verlag, 2002.

[26] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration illustrated by the Stormer/Verlet method*, Acta Numerica **12** (2003), 399–450.

[27] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations*, vol. 31, Springer-Verlag, 2006.

[28] E. Hairer and G. Wanner, *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, Springer, 1996.

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

[30] ———, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, Journal of Computational Physics archive **221** (2007), no. 4, 739–760.

[31] A.T Ihler, *An overview of fast multipole methods*, 2004.

[32] J.D. Jackson, *Classical electrodynamics: Third edition*, John Wiley and Sons, 1999.

[33] J. Jia and J. Huang, *Krylov deferred correction accelerated method of lines transpose for parabolic problems*, Journal of Computational Physics **227** (2008), no. 3, 1739–1753.

[34] J. Jia and J. Liu, *Stable and spectrally accurate schemes for the Navier-Stokes equations*, SIAM Journal on Scientific Computing **33** (2011), no. 5, 2421–2439.

[35] C.T. Kelly, *Iterative methods for linear and nonlinear equations*, SIAM, 1995.

[36] ———, *Solving nonlinear equations with Newton's method*, SIAM, 2003.

[37] D.A. Knoll and D.E. Keyes, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, Journal of Computational Physics **193** (2004), no. 2, 357 – 397.

[38] D. Kushnir and V. Rokhlin, *A highly accurate solver for stiff ordinary differential equations*, SIAM Journal of Scientific Computation **34** (2012), no. 3, A1296–A1315.

[39] A.T. Layton and M. Minion, *Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics*, Journal of Computational Physics **194** (2004), 697–714.

[40] R. Leveque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, SIAM, 2007.

[41] S. Li and L.R. Petzold, *Software and algorithms for sensitivity analysis of large-scale differential algebraic systems*, Journal of Computational and Applied Mathematics **125** (2000), no. 1, 131–145.

[42] J. Lions, Y. Maday, and G. Turinici, *Resolution d'EDP par un schema en temps pararee*l, Comptes Rendus de l'Academie des Sciences **332** (2001), 661–668.

[43] B. Lu, C. Xiaolin, J. Huang, and A.J. McCammon, *Order N algorithm for computation of electrostatic interactions in biomolecular systems*, Proceedings of the National Academy of Sciences **103** (2006), no. 51, 19314–19319.

[44] F. Mazzia et al., *Test set for IVP solvers*, `https://www.dm.uniba.it/~testset/testsetivpsolvers`.

[45] P.S. Pacheco, *An introduction to parallel programming*, Elsvier Inc, Boston, MA., 2011.

[46] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM Journal of Scientific and Statistical Computing **7** (1986), 856–869.

[47] V. Simoncini and D.B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM Journal on Scientific Computing **25** (2003), 454–477.

[48] R. Spec, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon, *A massively space-time parallel n-body solver*, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Los Alamitos, CA, USA), SC '12, IEEE Computer Society Press, 2012, pp. 92:1–92:11.

[49] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, Springer, 1992.

[50] L.N. Trefethen, *Is Gauss quadrature better than Clenshaw-Curtis?*, SIAM Review **50** (2008), no. 1, 67–87.

[51] L.N. Trefethen and D. Bau, *Numerical linear algebra*, SIAM, Philadelphia, PA., 1997.

[52] B. Zhang, *Integral-equation-based fast algorithms and graph-theoretic methods for large-scale simulations*, Ph.D. thesis, University of North Carolina at Chapel Hill, 2010.