Towards Improving Binary Black Hole Simulations Using Compact Finite Differences

James Bleazard

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Eric Hirschmann, Advisor

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Towards Improving Binary Black Hole Simulations Using Compact Finite Differences

James Bleazard
Department of Physics and Astronomy, BYU
Bachelor of Science

Binary black hole simulations will need increased waveform accuracy in the next few years as next generation gravitational wave observatories come online. Current accuracy of numerical relativity codes is sufficient for today's gravitational wave detections. But observatories such as LISA may require an increase in accuracy of up to two orders of magnitude. Compact Finite Differences (CFDs) have been used increasingly in a variety of engineering applications with a corresponding increase in accuracy. We are working to incorporate compact finite differencing methods in evolution codes for general relativity. Merger simulations with large mass ratios, spins, eccentricities and charges require significantly more computational resources than current evolutions can manage. By using CFDs, we hope to decrease computational times while improving the accuracy. These methods are ultimately implicit schemes but do not dramatically increase computational cost because we can precalculate matrix elements. We have developed a means for calculating these operators based on the order of the derivative, the order of accuracy, and the nature of the banded matrix. An issue that has recently drawn attention is ensuring that the resulting operators are stable. We report on our efforts to find stable, highly accurate schemes of first and second derivatives.

Keywords: Accuracy, Binary Black Hole, Mergers, Compact Finite Differencing, Dendro-GR, Numerical Modeling, Numerical Stability

ACKNOWLEDGMENTS

I would like to gratefully acknowledge funding from the National Science Foundation through grant NSF PHY-2207615. I would also like to thank Dr. Eric Hirschmann, Dr. David Nielsen and Dr. William Black for their mentorship and guidance throughout this project. Lastly, I would like to thank Brigham Young University and the College of Computation, Physical and Mathematical Sciences for making this project possible, alongside my group members Nathanael Garey and Luke Papenfuss for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As gravitational wave detectors advance, a better understanding of the sources of these gravitational waves is required. Current numerical models provide enough information about these sources for current detectors, but future detectors are estimated to observe more complex systems which are very computationally expensive to model. We explore the possibility of using compact finite differencing to improve these simulations.

1.1 Future Black Hole Binary Models

When two massive and compact objects collide, they bend the fabric of spacetime so significantly, they create a gravitational wave that we can measure [1]. These compact objects are most commonly black holes, but they can also involve similarly dense objects like neutron stars. From these gravitational waves, we can learn about many of the properties of the colliding objects with gravitational wave detectors. Over the next few decades, multiple gravitational wave detectors will be built. These detectors will observe at lower and higher frequencies than current detectors and will be more precise.

In order to understand the gravitational wave signals received, researchers require template signals to compare against [2]. These example signals are generated using numerical relativity codes. These codes simulate the events that cause gravitational waves with a variety of different physical conditions in order to provide a range of gravitational waves to compare against.

The first successful simulation of a binary black hole system using numerical relativity was created multiple decades ago. There have been more complex simulations made ever since then [3]. These simulations seek to calculate accurate waveforms of the gravitational waves the binary black hole mergers create. Libraries of template waveforms have been generated, covering much of the parameter space. However, there are gaps in this coverage. These gaps include more complex systems that will be potentially observable using new detectors [2].

Binary black hole systems with a high mass ratio have presented difficulties for simulations in the past and continue to do so. A few models have been created where the black holes have a mass ratio of up to 128 : 1, and one head on collision with a mass ratio of 1024 : 1 [4]. However, many more are needed to even begin to cover the parameter space. These models require very high resolution to resolve the smaller black hole on the grid, and to therefore produce an accurate waveform [2]. These models are important as they represent a source of low frequency signals detectable by future detectors such as LISA.

Other factors can increase the complexity of binary black hole models [2]. Systems with high spin require more complicated initial data, and have higher resolution requirements. Large eccentricity in the orbits of the black holes adds another dimension to the parameter space, and leads to increased complexity of the orbits. Lastly, effects like charged black holes and accretion disks add more parameters to the parameter space, and require more complex equations to simulate. Various attempts to simulate all these effects have been done. However, attempting to simultaneously implement several of these effects requires substantial computational resources and will ultimately require more accurate and efficient simulation techniques.

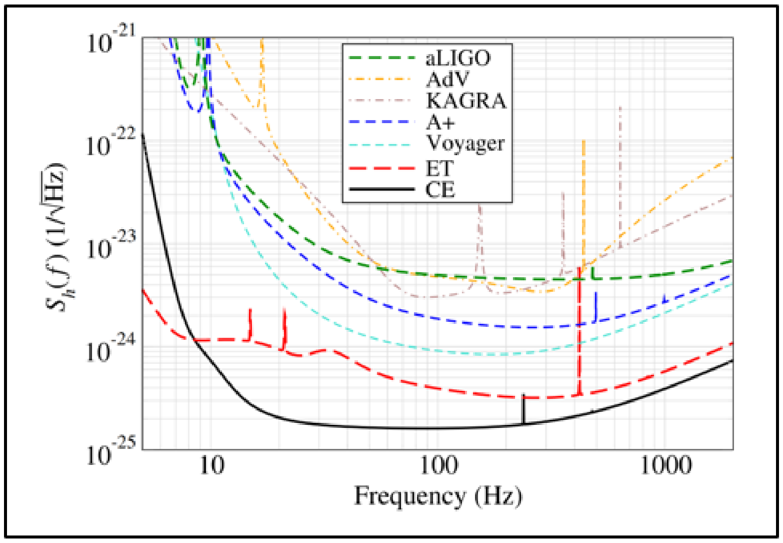

Figure 1.1 Sensitivity of future gravitational wave detectors from [5]. This plot depicts the designed power spectrum density of each gravitational wave detector. These are predicted values of the sensitivity of each detector based on frequency. Each line depicts the power spectrum density of a different detector, which each use interferometer technology to detect gravitational waves. Much of the current gravitational wave data comes from Advanced LIGO (aLIGO). The Einstein Telescope (ET) and Cosmic Explorer (CE) are planned detectors, with a predicted increase in sensitivity of more than an order of accuracy across the frequency spectrum.

Lastly, future detectors will be more sensitive, as shown in Figure 1.1. This means incoming data will have signal to noise ratios in the thousands with multiple simultaneous signals. Current simulations do not yet provide sufficient accuracy in their calculated waveforms to match these sensitivity levels [6]. More accurate waveforms will allow for better comparison with these signals, and easier decoupling of simultaneous signals.

If the numerical relativity community can increase the accuracy of numerical relativity models without dramatically increasing the computational cost, more complex models and higher accuracy can be achieved. One possible technique that may help us reach this goal may be using Compact Finite Differences.

1.2 Compact Finite Differencing

Currently, our numerical relativity group at Brigham Young University uses Dendro-GR to simulate binary black hole mergers and collect the waveforms they generate [7]. Dendro-GR employs many techniques to decrease the high numerical cost of binary black hole simulations without sacrificing accuracy. Namely, Wavelet Adaptive Mesh Refinement is used to dynamically change the resolution across the space [8]. This requires the use of many smaller grid sections, and therefore boundary closures, that can form a larger grid for simulation. We are working towards the goal of using Compact Finite Difference (CFD) techniques to increase the accuracy of these simulations without increasing the required resolution of the grid [9].

Compact Finite Differencing schemes (CFDs) have been used in a variety of engineering applications to increase the accuracy of these simulations [10]. Primarily, CFDs have been used to improve computational aeroacoustics codes [11]. They have also shown success in stable, high-order, nonlinear pressure waves [12]. These models showed both increased accuracy and stability without a corresponding increase in grid refinement. By using the same resolution, these codes keep a consistent computational cost, whilst also increasing the accuracy of the simulation. We hope that these results can be replicated when applied to binary black hole models.

Most current numerical relativity codes, including Dendro-GR, use explicit schemes to approximate derivatives and solutions to differential equations. These schemes calculate the derivative at a single point using the function value at nearby points. CFDs are implicit schemes, using nearby function and nearby derivative values to approximate the derivative at a point. This differs from traditional explicit schemes which only use nearby function points to approximate the derivative. Such implicit schemes require matrix calculations, and traditionally a matrix inversion would need to be performed at every time-step. Using CFDs, this matrix evolution can be performed once and reused, allowing computational costs to remain constant with resolution [9]. The use of these derivative values allows for an increase in order of accuracy without needing higher resolution.

This higher order of accuracy also extends to the boundaries of the spatial volume. By better representing derivatives on the boundary, CFDs can potentially achieve orders of magnitude improvement in accuracy as compared to explicit schemes. Dendro-GR relies on the use of many boundary closures across the domain, as each of the borders between smaller grids require boundaries, as well as the boundaries at the edges of the simulation. An increase in accuracy here would allow for a large increase in overall accuracy of the simulation. This is why we believe CFDs present an exciting possibility for improving Dendro-GR.

Lastly, we hope to also replicate similar findings to those found by Kim [11] and Sharan, Brady and Livescu [13] to further increase stability and accuracy of these schemes. They use different optimization techniques to improve on the standard schemes.

Chapter 2

Methods

We can approximate a derivative based on its function value using CFDs. Doing so produces systems of equations relating the function values to their derivatives at discrete points. By calculating the coefficients for each term we can set an order of accuracy for these systems and use them to make approximations. We cover how to calculate these coefficients and a Mathematica script we made to automate the process.

2.1 Analytic CFD Operators

One approach to numerical differentiation is to use finite difference schemes to approximate the derivative of a function across a grid. Different schemes provide different levels of accuracy and stability depending on the number of points across the domain. Compact finite differencing employs the use of implicit schemes, as opposed to the explicit schemes traditionally used, in an attempt to increase accuracy without increasing the required number of points across the domain. We explore how to set up these schemes. Much of this work can be found in the excellent thesis by Tyler [9].

2.1.1 Explicit vs. Implicit Schemes

A traditional explicit scheme uses the function value from nearby points to approximate a derivative, often at a central point. It estimates the derivative value at the central point by finding the slope of a line made from the nearby points. The following scheme is a centered difference scheme with order of accuracy $\mathscr{O}(h^2)$, or second order. This means that the error can be approximated by the spacing between the grid points, h, squared.

$$\phi_i' = \frac{1}{2h}(\phi_{i+1} - \phi_{i-1}) + \mathscr{O}(h^2) \tag{2.1}$$

In contrast, an implicit scheme also uses derivative values at surrounding points to perform the approximation. Implicit schemes thus create a system of equations which can be solved to approximate the derivative value at every point. The following implicit scheme is known as a tridiagonal scheme, as the left hand side only has three terms.

$$\frac{1}{4}\phi_{i+1}' + \phi_i' + \frac{1}{4}\phi_{i-1}' = \frac{3}{4h}(\phi_{i+1} - \phi_{i-1}) + \mathscr{O}(h^4) \tag{2.2}$$

Importantly, the above scheme uses values from the same number of points as the scheme in Equation 2.1, but is fourth order, rather than second order.

Implicit schemes can be created for any derivative order. Still, our main focus is first and second derivatives, as those make up the derivatives within the Einstein equations that we use to simulate binary black hole mergers. The Einstein equations define spacetime within general relativity. The following equation is an example of a possible second derivative scheme:

$$\frac{1}{3}\phi_{i+1}'' + \phi_i'' + \frac{1}{3}\phi_{i-1}'' = \frac{14}{9h^2}(\phi_{i+1} - 2\phi_i + \phi_{i-1}) + \frac{1}{81h^2}(\phi_{i+2} - 2\phi_i + \phi_{i-2}) + \mathscr{O}(h^6) \tag{2.3}$$

2.1.2 Interior Scheme Coefficients

The schemes discussed in Sections 2.1.1 are known as interior schemes as they do not work to approximate the derivative on the boundary. The boundary will be discussed more in 2.1.3. An example of a pentadiagonal, eighth-order scheme is as follows:

$$\beta\phi'_{i+2}+\alpha\phi'_{i+1}+\phi'_i+\alpha\phi'_{i-1}-\beta\phi'_{i+2}=\frac{a_1}{2h}(\phi_{i+1}-\phi_{i-1})+\frac{a_2}{4h}(\phi_{i+2}-\phi_{i-2})+\frac{a_3}{6h}(\phi_{i+3}-\phi_{i-3}) \tag{2.4}$$

Where β, α, a_1, a_2, and a_3 are arbitrary coefficients that we will calculate.

To calculate each of the coefficients in this scheme, we Taylor expand each function and derivative, then group in powers of the spacing, h, creating a system of equations. The Taylor expansion for the derivatives are as follows:

$$\phi'_{i-2}=\phi'_i-2h\phi''_i+\frac{2^2h^2}{2!}\phi'''_i-\frac{2^3h^3}{3!}\phi'''_i+\cdots \tag{2.5}$$

$$\phi'_{i-1}=\phi'_i-h\phi''_i+\frac{h^2}{2!}\phi'''_i-\frac{h^3}{3!}\phi'''_i+\cdots \tag{2.6}$$

$$\phi'_i=\phi'_i \tag{2.7}$$

$$\phi'_{i+1}=\phi'_i+h\phi''_i+\frac{h^2}{2!}\phi'''_i+\frac{h^3}{3!}\phi'''_i+\cdots \tag{2.8}$$

$$\phi'_{i+2}=\phi'_i+2h\phi''_i+\frac{2^2h^2}{2!}\phi'''_i+\frac{2^3h^3}{3!}\phi'''_i+\cdots \tag{2.9}$$

The Taylor expansion for the function values are as follows:

$$\phi_{i-3} = \phi_i - 3h\phi_i' + \frac{3^2h^2}{2!}\phi_i'' - \frac{3^3h^3}{3!}\phi_i''' + \cdots \tag{2.10}$$
$$\phi_{i-2} = \phi_i - 2h\phi_i' + \frac{2^2h^2}{2!}\phi_i'' - \frac{2^3h^3}{3!}\phi_i''' + \cdots \tag{2.11}$$
$$\phi_{i-1} = \phi_i - h\phi_i' + \frac{h^2}{2!}\phi_i'' - \frac{h^3}{3!}\phi_i''' + \cdots \tag{2.12}$$
$$\phi_{i+1} = \phi_i + h\phi_i' + \frac{h^2}{2!}\phi_i'' + \frac{h^3}{3!}\phi_i''' + \cdots \tag{2.13}$$
$$\phi_{i+2} = \phi_i + 2h\phi_i' + \frac{2^2h^2}{2!}\phi_i'' + \frac{2^3h^3}{3!}\phi_i''' + \cdots \tag{2.14}$$
$$\phi_{i+3} = \phi_i + 3h\phi_i' + \frac{3^2h^2}{2!}\phi_i'' + \frac{3^3h^3}{3!}\phi_i''' + \cdots \tag{2.15}$$

By plugging all of these Taylor expansions into Equation 2.4 and isolating terms with the same order of derivative, we can create the system of equations. These can then be solved to calculate each of the coefficients, which can then be plugged back into the original scheme. This system of equations is as follows:

$$2(\alpha+\beta)+1 = a_1 + a_2 + a_3 \qquad \mathcal{O}(h^2) \tag{2.16}$$
$$2\frac{3!}{2!}(\alpha+2^2\beta) = a_1 + 2^2a_2 + 3^2a_3 \qquad \mathcal{O}(h^4) \tag{2.17}$$
$$2\frac{5!}{4!}(\alpha+2^4\beta) = a_1 + 2^4a_2 + 3^4a_3 \qquad \mathcal{O}(h^6) \tag{2.18}$$
$$2\frac{7!}{6!}(\alpha+2^6\beta) = a_1 + 2^6a_2 + 3^6a_3 \qquad \mathcal{O}(h^8) \tag{2.19}$$
$$2\frac{9!}{8!}(\alpha+2^8\beta) = a_1 + 2^8a_2 + 3^8a_3 \qquad \mathcal{O}(h^{10}) \tag{2.20}$$
$$\vdots \tag{2.21}$$

We can generate an infinite number of equations with this method, but can only use five to solve for the coefficients in the above scheme. Solving the five equations above would net a tenth-order interior scheme with five coefficients. However we could choose to solve only three of the equations above, generating a fourth-order scheme with two arbitrary parameters we can optimize over. We call these arbitrary parameters free parameters, as they are left unconstrained by the above equations. Also, more equations can be solved to create a scheme with a higher order of accuracy, but then the scheme must be expanded to include more coefficients such that the system is not overdetermined. The reader can do this calculation, but an automated method of doing this calculation for arbitrary schemes is discussed in Section 2.4.

2.1.3 Boundary Scheme Coefficients

As Figure 2.1 displays, a different scheme must be used to approximate derivatives on and near boundaries. These schemes are called boundary closure schemes. Due to the lack of points on one side of the boundary, it is impossible to use a centered derivative scheme. To compensate for the lack of points to one side, more points are used to accurately approximate the derivative.

Different schemes must be used for each point near the boundary. Depending on the order of accuracy and the format of the interior scheme, different near-boundary points will require different boundary closure schemes. In principle, these boundary closure schemes can also have different orders of accuracy, can rely on different numbers of points, and can be used to accommodate physical boundary conditions so there is considerable possible arbitrariness.

For these schemes, we will be using a grid of $N+1$ points, starting at point 0 to point N, with spacing h. In these schemes, we use γ to represent arbitrary coefficients, with their subscripts denoting where in the final coefficient matrix they belong. An example of a scheme expressed using the coefficient matrices is found in Section 2.1.4. A possible set of schemes for points around the boundary are given below.

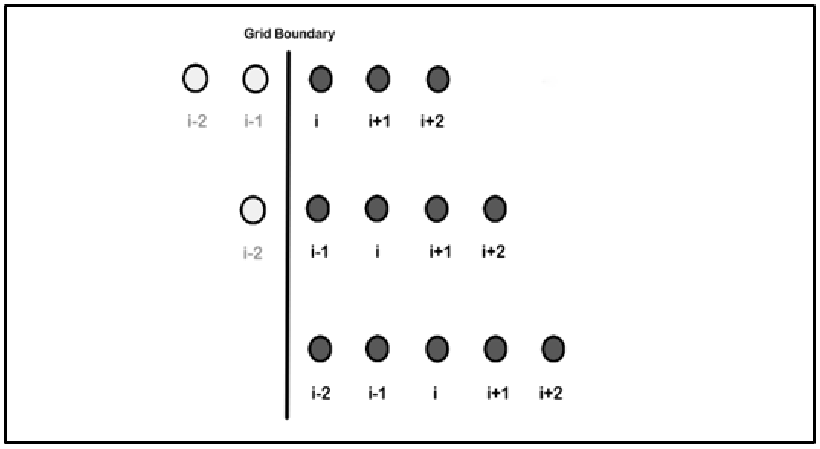

Figure 2.1 Need for Boundary Closure Schemes. Our basic requirement is to solve for the derivative at every point in the domain. As such. in the interior of our domain of interest schemes tend to be be centered around each point. This becomes problematic when the points lie on the edge of the domain. The figure depicts the points available to a centered scheme at each of the points closest to the boundary. The filled in points are available, and the open points are not. This means we must rely exclusively on points to the right of the boundary to approximate the derivative, requiring a change to the scheme in the vicinity of boundaries.

Boundary Point 0 First Derivative:

$$\phi_0' + \gamma_{01}\phi_1' + \gamma_{02}\phi_2' = \frac{1}{h}\sum_{i=0}^{k} a_{0i}\phi_i \tag{2.22}$$

Boundary Point 1 First Derivative:

$$\gamma_{10}\phi_0' + \phi_1' + \gamma_{12}\phi_2' + \gamma_{13}\phi_3' = \frac{1}{h}\sum_{i=0}^{k} a_{1i}\phi_i \tag{2.23}$$

Boundary Point 2 First Derivative:

$$\gamma_{20}\phi_0' + \gamma_{21}\phi_1' + \phi_2' + \gamma_{23}\phi_3' + \gamma_{24}\phi_4' = \frac{1}{h}\sum_{i=0}^{k} a_{2i}\phi_i \tag{2.24}$$

Boundary Point N-2 First Derivative:

$$\gamma_{20}\phi'_N + \gamma_{21}\phi'_{N-1} + \phi'_{N-2} + \gamma_{23}\phi'_{N-3} + \gamma_{24}\phi'_{N-4} = -\frac{1}{h}\sum_{i=0}^{k} a_{2i}\phi_{N-i} \tag{2.25}$$

Boundary Point N-1 First Derivative:

$$\gamma_{10}\phi'_N + \phi'_{N-1} + \gamma_{12}\phi'_{N-2} + \gamma_{13}\phi'_{N-3} = -\frac{1}{h}\sum_{i=0}^{k} a_{1i}\phi_{N-i} \tag{2.26}$$

Boundary Point N First Derivative:

$$\phi'_N + \gamma_{01}\phi'_{N-1} + \gamma_{02}\phi'_{N-2} = -\frac{1}{h}\sum_{i=0}^{k} a_{Ni}\phi_{N-i} \tag{2.27}$$

In each of the equations above, N denotes the number of intervals contained within the grid, and k is an integer that takes values between 1 and N. Increasing k can increase the order of accuracy of each of these schemes, it also increases the number of parameters on the right hand side. These schemes can be modified to accommodate periodic boundary conditions, or other types of boundary conditions.

2.1.4 Matrix Representations

Now that we have an equation for each point across the grid, we can set up a system of equations for each derivative. The following describes the basic setup for the matrix equation for the derivatives.

$$P\phi' = Q\phi \tag{2.28}$$

Where P and Q are coefficient matrices, ϕ' is a vector of the derivative values, and ϕ is a vector of the function values.

A full version of the previous equation in terms of components is found in the following matrix equation. This particular version uses a sixth-order pentadiagonal scheme with two boundary closure rows.

$$\begin{pmatrix} 1 & \gamma_{01} & \gamma_{02} & 0 & 0 & \cdots & 0 & 0 \\ \gamma_{10} & 1 & \gamma_{12} & \gamma_{13} & 0 & \cdots & 0 & 0 \\ \beta & \alpha & 1 & \alpha & \beta & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta & \alpha & 1 & \alpha & \beta \\ 0 & 0 & \cdots & 0 & \gamma_{13} & \gamma_{12} & 1 & \gamma_{10} \\ 0 & 0 & \cdots & 0 & 0 & \gamma_{02} & \gamma_{01} & 1 \end{pmatrix} \begin{pmatrix} \phi'_0 \\ \phi'_1 \\ \phi'_2 \\ \vdots \\ \phi'_{N-2} \\ \phi'_{N-1} \\ \phi'_N \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & 0 & \cdots & 0 \\ a_{10} & a_{11} & a_{12} & a_{13} & 0 & \cdots & 0 \\ \frac{a_2}{4h} & -\frac{a_1}{2h} & 0 & \frac{a_1}{2h} & \frac{a_2}{4h} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{a_2}{4h} & -\frac{a_1}{2h} & 0 & \frac{a_1}{2h} & \frac{a_2}{4h} \\ 0 & \cdots & 0 & -a_{13} & -a_{12} & -a_{11} & -a_{10} \\ 0 & \cdots & 0 & -a_{03} & -a_{02} & -a_{01} & -a_{00} \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-2} \\ \phi_{N-1} \\ \phi_N \end{pmatrix} \tag{2.29}$$

The above matrices illustrate the difference between the interior schemes and the boundary schemes. The top and bottom two rows are boundary schemes. We can see that they require a specific, asymmetric form because there are not enough points to use a central scheme. The boundary rows are also different schemes from each other, unlike the interior scheme. The interior scheme is symmetric across the diagonal in both matrices, and follows a consistent form across all rows.

2.2 Spectral Function

CFD schemes, such as the one expressed Equation 2.4, are inherently approximations. One possible way to represent this is by adding a term accounting for the order of error of the approximation, $\mathcal{O}$. This is shown below.

$$\beta\phi'_{i+2} + \alpha\phi'_{i+1} + \phi'_i + \alpha\phi'_{i-1} - \beta\phi'_{i+2} = \frac{a_1}{2h}(\phi_{i+1} - \phi_{i-1}) + \frac{a_2}{4h}(\phi_{i+2} - \phi_{i-2}) + \frac{a_3}{6h}(\phi_{i+3} - \phi_{i-3}) + \mathcal{O}(h^{10}) \tag{2.30}$$

Just by adding the extra term we don't get much in the way of additional analysis. Another method of representing the inherent approximation is to replace the ϕ' terms with $\bar{\phi}'$ terms. Here $\bar{\phi}$ is not the same function as ϕ, but the new function is introduced to satisfy the equality and will likely be similar to the original function. The previous scheme with this substitution made becomes:

$$\beta\bar{\phi}'_{i+2} + \alpha\bar{\phi}'_{i+1} + \bar{\phi}'_i + \alpha\bar{\phi}'_{i-1} - \beta\bar{\phi}'_{i+2} = \frac{a_1}{2h}(\phi_{i+1} - \phi_{i-1}) + \frac{a_2}{4h}(\phi_{i+2} - \phi_{i-2}) + \frac{a_3}{6h}(\phi_{i+3} - \phi_{i-3}) \tag{2.31}$$

Now consider doing a Fourier analysis by taking the Fourier transform of this scheme. A Fourier transform allows us to move to frequency space. This will allow us to examine the scheme across wavenumbers which range from 0 to π. Using the following relations, we can perform the Fourier transform.

$$\phi_{i+s} = \tilde{\phi}_i \exp\left(\frac{2\pi ikhs}{L}\right) \qquad \bar{\phi}_{i+s} = \tilde{\bar{\phi}}_i \frac{2\pi ikh}{L} \exp\left(\frac{2\pi ikhs}{L}\right) \tag{2.32}$$

We can plug these relations into our scheme, and simplify to find the following scheme, which has been converted into Fourier space.

$$kh \cdot \tilde{\bar{\phi}}(k)\left[2\beta\cos(2kh) + 2\alpha\cos(kh) + 1\right] = \tilde{\phi}(k) \cdot \left[a_1 \sin(kh) + \frac{a_2}{2}\sin(2kh) + \frac{a_3}{3}\sin(3kh)\right]. \tag{2.33}$$

The equation above can be simplified by first substituting κ for kh, then moving the $\tilde{\phi}$ and $\tilde{\bar{\phi}}$ to the same side. This simplification is shown below.

$$\bar{\kappa}(\kappa) \equiv \kappa \cdot \frac{\tilde{\bar{\phi}}(k)}{\tilde{\phi}(k)} = \frac{a_1 \sin(\kappa) + \frac{a_2}{2}\sin(2\kappa) + \frac{a_3}{3}\sin(3\kappa)}{1 + 2\alpha\cos(\kappa) + 2\beta\cos(2\kappa)} \tag{2.34}$$

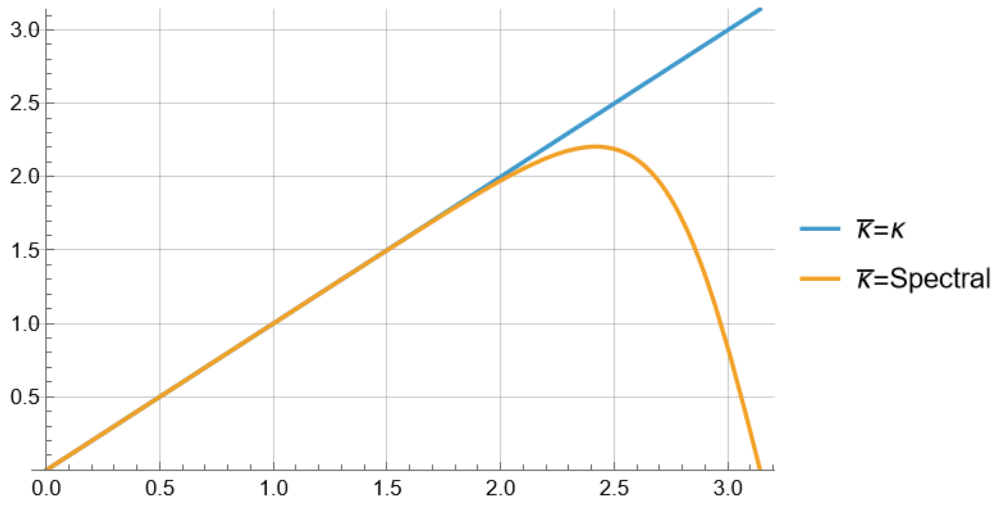

Figure 2.2 Eighth-order interior pentadiagonal spectral function example. This plot shows the comparison between a linear spectral function and the spectral function in Equation 2.34. The blue line is the linear spectral function we are striving for, whereas the orange line shows the spectral function of the scheme we've been exploring. As we can see, the orange curve follows the linear curve very well for small wavenumbers, but diverges at high wavenumbers. We can optimize the parameters in our schemes to better fit our spectral function, and improve the spectral accuracy.

$\bar{\kappa}$ is known as the spectral function, or the transfer function for the scheme in Equation 2.4. This equation can be used to measure the quality of the approximation we made in Equation 2.31. Figure 2.2 shows an example of this comparison. If $\bar{\kappa}(\kappa)$ is linear, then the approximation is exactly the same as the original function. For first derivatives, this is impossible as the spectral function has the condition that at $\kappa = \pi$, the spectral function equals zero.

2.3 Filtering

Many CFD schemes are unstable and can cause large errors to appear across a grid as a simulation proceeds. But, some of these unstable schemes can be made highly accurate and have the potential to further improve numerical simulations if they can be rendered stable. By filtering the function values after applying a form of the CFD scheme we can apply dissipation, thereby reducing the noise on the grid. Below is an example of such a filtering scheme:

$$R\hat{\phi} = S\phi \tag{2.35}$$

Where R and S, like the P and Q operators, are coefficient matrices, ϕ is the vector of function values, and $\hat{\phi}$ is the filtered function values. Filtering schemes of this form can be applied to standard CFD schemes as follows.

$$P\phi' = Q(R^{-1}S)\phi \tag{2.36}$$

The following is an example of an interior filtering scheme that complements the first derivative schemes discussed throughout Section 2.1.2.

$$\beta_f\hat{\phi}_{i-2} + \alpha_f\hat{\phi}_{i-1} + \hat{\phi}_i + \alpha_f\hat{\phi}_{i+1} + \beta_f\hat{\phi}_{i+2} = \sum_{k=0}^{M}\frac{a_k}{2}(\phi_{i+k} + \phi_{i-k}) \tag{2.37}$$

Where M corresponds with the order of the filter being generated. In practice, most filters used are at least two orders higher than the CFD scheme they are coupled with.

In order to solve for the coefficients in this scheme, we need a series of constraint equations similar to those used when generating the CFD schemes. The first constraint we can use comes from the spectral function of this scheme, shown below.

$$\bar{\kappa}(\kappa) = \frac{\sum_{k=0}^{M} a_k\cos(k\kappa)}{1 + 2\alpha_f\cos(\kappa) + 2\beta_f\cos(2\kappa)} \tag{2.38}$$

We need this spectral function to match the interior spectral functions of the compact schemes, meaning that at wavenumber π, we need the spectral function to become zero. This results in the following constraint equation.

$$\sum_{k=0}^{M}(-1)^k a_k = 0 \tag{2.39}$$

The rest of the required constraint equations can be derived from the Taylor series expansion of the scheme. This is slightly more complex than the traditional scheme as the $\hat{\phi}$ terms are not derivatives, rather they are filtered values of the original function. Therefore their Taylor expansion is different, and given below

$$\hat{\phi}_{i+k} = \phi_i + kh\phi_i' + \frac{k^2h^2}{2!}\phi'' + \cdots + \frac{k^nh^n}{n!}\left(\frac{\partial^n\phi}{\partial x^n}\right)_i + R_n(x) \tag{2.40}$$

where

$$R_n(x) = \frac{k^{n+1}h^{n+1}}{(n+1)!}\left(\frac{\partial^{n+1}\phi}{\partial x^{n+1}}\right)_{i+\xi} \tag{2.41}$$

We then plug this expansion into the previous scheme, alongside the traditional Taylor expansion for all of the ϕ terms, and isolate new equations for each power of h. This allows us to generate the rest of the constraint equations for the desired order of accuracy. The first five of these equations are shown below.

$$\sum_{k=0}^{M} a_k = 2\alpha_f + 2\beta_f + 1 \tag{2.42}$$

$$\sum_{k=1}^{M} k^2 a_k = 2\alpha_f + 2^3\beta_f \tag{2.43}$$

$$\sum_{k=1}^{M} k^4 a_k = 2\alpha_f + 2^5\beta_f \tag{2.44}$$

$$\sum_{k=1}^{M} k^6 a_k = 2\alpha_f + 2^7\beta_f \tag{2.45}$$

$$\sum_{k=1}^{M} k^8 a_k = 2\alpha_f + 2^9\beta_f \tag{2.46}$$

Now, we have all the tools required to solve for each coefficient in the filtering schemes. Traditionally though, we choose to leave α_f and β_f general so that they can be optimized later to best improve whichever scheme the filter is being applied to. But, these values are limited by the spectral function in Equation 2.38. As the following appears in the denominator of the spectral function, it must be greater than zero when κ is between 0 and π.

$$1 + 2\alpha_f \cos(\kappa) + 2\beta_f \cos(2\kappa) > 0 \tag{2.47}$$

This means that for tridiagonal filters, where β_f is zero, α_f can range from -0.5 to 0.5. But, as should be clear, the pentadiagonal case is more complicated.

Just like the CFD schemes, the filtering schemes also require special attention on the boundaries. This requires shifted stencils to account for the lack of points to maintain a centered scheme, like in Figure 2.1. The specific boundary schemes for these filters are simpler than those for traditional CFD schemes. Below are the schemes for the first four boundary nodes. As the order of accuracy increases, M will increase, adding more parameters.

$$\hat{\phi}_1 + \alpha_f \hat{\phi}_2 + \beta_f \hat{\phi}_3 = \sum_{k=1}^{M+1} a_k \phi_k \tag{2.48}$$

$$\alpha_f \hat{\phi}_1 + \hat{\phi}_2 + \alpha_f \hat{\phi}_3 + \beta_f \hat{\phi}_4 = \sum_{k=1}^{M+1} a_k \phi_k \tag{2.49}$$

$$\beta_f \hat{\phi}_1 + \alpha_f \hat{\phi}_2 + \hat{\phi}_3 + \alpha_f \hat{\phi}_4 + \beta_f \hat{\phi}_5 = \sum_{k=1}^{M+1} a_k \phi_k \tag{2.50}$$

$$\beta_f \hat{\phi}_2 + \alpha_f \hat{\phi}_3 + \hat{\phi}_4 + \alpha_f \hat{\phi}_5 + \beta_f \hat{\phi}_6 = \sum_{k=1}^{M+1} a_k \phi_k \tag{2.51}$$

Similarly, the following equations are the boundary schemes for the last four nodes, nodes $N-3$ through N.

$$\beta_f \hat{\phi}_{N-1} + \alpha_f \hat{\phi}_{N-2} + \hat{\phi}_{N-3} + \alpha_f \hat{\phi}_{N-4} + \beta_f \hat{\phi}_{N-5} = \sum_{k=N-M}^{N} a_k \phi_k \tag{2.52}$$

$$\beta_f \hat{\phi}_{N} + \alpha_f \hat{\phi}_{N-1} + \hat{\phi}_{N-2} + \alpha_f \hat{\phi}_{N-3} + \beta_f \hat{\phi}_{N-4} = \sum_{k=N-M}^{N} a_k \phi_k \tag{2.53}$$

$$\alpha_f \hat{\phi}_{N} + \hat{\phi}_{N-1} + \alpha_f \hat{\phi}_{N-2} + \beta_f \hat{\phi}_{N-3} = \sum_{k=N-M}^{N} a_k \phi_k \tag{2.54}$$

$$\hat{\phi}_{N} + \alpha_f \hat{\phi}_{N-1} + \beta_f \hat{\phi}_{N-2} = \sum_{k=N-M}^{N} a_k \phi_k \tag{2.55}$$

To find the coefficients for these schemes, one can use the spectral function and the Taylor expansion in the same process as the interior scheme. Importantly, the α_f and β_f values in these boundary schemes are the same values as those in the interior scheme.

2.4 Generating Operators

We have developed a Mathematica script to automate the process for determining and solving for the coefficients using the structure outlined by Tyler [9], and found in our Section 2.1. This script takes user input to determine the parameters associated with the particular scheme we want to derive. It then sets up the matrices corresponding to these properties, creates a system of equations for the matrix coefficients, solves this system, and outputs the coefficients in the final scheme. This process takes a few seconds before outputting values for these matrices.

2.4.1 User Input

When using the script to develop operators, users can choose between many options to suit their needs. First and foremost, the user can select between first or second derivative, the desired order of accuracy, and the structure of the matrix. Specifically, the matrices output can use either tridiagonal or pentadiagonal interior schemes. Then users can choose the order of accuracy on the boundaries.

Alongside these choices, the user can choose to expand the number of boundary rows or expand the stencil size, adding extra terms to each equation. Each of these coefficients can then be chosen to be fixed or free. Fixed coefficients are solved for normally and will output as a number or equation in terms of free parameters. Free parameters stay symbolic throughout the solve, allowing the user to vary them later to hopefully increase accuracy or stability. Input may be given through dialog boxes that open throughout runtime, or a parameter file loaded in prior to the code being run.

2.4.2 Setting up Matrices

Once the user has made the necessary selections, the script generates symbolic versions of the operators whose coefficients will be found. These are stored for later use, and to provide context for the equations being solved. One example of code being used to generate these symbolic matrices, and its output is below. It depicts the construction of a tridiagonal matrix with no boundary closures as demonstration. The matrix is generated at runtime, and a user could easily change its size.

We set an adjustable size for the matrices being developed in order to adapt to all necessary parameters, and easily display the full structure of the matrix. In particular, at least three interior rows are shown to ensure the full boundary closure scheme is visible in the output, and the interior scheme can be contextualized in place within the operator.

2.4.3 Solving for Coefficients

Once the matrices are generated, the script collects all the parameters to be solved for into individual lists. These lists delineate coefficients that can be solved for by a specific set of equations. They also differentiate which coefficients will go into solving for the boundary closures, or the interior region of the operator. Lastly, they collect the coefficients that have been chosen to be free parameters to ensure every other coefficient has the correct dependence on them.

We calculated the requisite Taylor series for each derivative, as in Section 2.1, such that they can be separated into enough equations for each order of accuracy, and the number of coefficients to be solved for. We set a minimum order of accuracy, as the lower orders have less general systems of equations, but the dialog boxes communicate this minimum order, ensuring users are aware of it. Beyond this minimum order, all equations are determined automatically by leveraging patterns that appear. Finally, Mathematica's built in Solve function is used to solve each of these systems for their required variables and store the expressions produced.

2.4.4 Output

Upon completing the earlier steps, the Mathematica script outputs the information stored throughout the runtime. Starting with the symbolic versions of both matrices so that the user is very aware of the structure of the final operators. Then lists are output of free parameters within both the interior and boundary closures. These lists allow the user to set these parameters en masse when later implementing the operators. Finally, the script outputs code in a variety of coding languages and formats, setting each parameter to the calculated value. These final values are always fractions to preserve the precision, and automatically set as doubles. Their names also match those in the output matrices to minimize confusion. The user must then set up these operators within their own environment and plug in the coefficients. This is important so that the user can determine the data structure used to create the operators to fully optimize their code.

Below is an example output of the Mathematica code with the corresponding scheme in equations 2.56 through 2.59, and final coefficients in Table 2.1.

Interior:

$$\alpha\phi'_{i+1} + \phi'_i + \alpha\phi'_{i-1} = \frac{a_1}{2h}(\phi_{i+1} - \phi_{i-1}) + \frac{a_2}{4h}(\phi_{i+2} - \phi_{i-2}) \tag{2.56}$$

First Boundary Row:

$$\phi_0' + \gamma_{01}\phi_1' = \sum_{i=0}^{3} a_{0i}\phi_i \tag{2.57}$$

Second Boundary Row:

$$\gamma_{10}\phi_0' + \phi_1' + \gamma_{12}\phi_2' = \sum_{i=0}^{3} a_{1i}\phi_i \tag{2.58}$$

Matrix Representation:

$$\begin{pmatrix} 1 & \gamma_{01} & 0 & 0 & \cdots & 0 & 0 \\ \gamma_{10} & 1 & \gamma_{12} & 0 & \cdots & 0 & 0 \\ 0 & \alpha & 1 & \alpha & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha & 1 & \alpha & 0 \\ 0 & 0 & \cdots & 0 & \gamma_{12} & 1 & \gamma_{10} \\ 0 & 0 & \cdots & 0 & 0 & \gamma_{01} & 1 \end{pmatrix} \overline{\phi'} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & 0 & \cdots & 0 \\ a_{10} & a_{11} & a_{12} & a_{13} & 0 & \cdots & 0 \\ \frac{a_2}{4h} & -\frac{a_1}{2h} & 0 & \frac{a_1}{2h} & \frac{a_2}{4h} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{a_2}{4h} & -\frac{a_1}{2h} & 0 & \frac{a_1}{2h} & \frac{a_2}{4h} \\ 0 & \cdots & 0 & -a_{13} & -a_{12} & -a_{11} & -a_{10} \\ 0 & \cdots & 0 & -a_{03} & -a_{02} & -a_{01} & -a_{00} \end{pmatrix} \overline{\phi} \tag{2.59}$$

a_1	a_2	a_{00}	a_{01}	a_{02}	a_{03}	a_{10}	a_{11}	a_{12}	a_{13}	γ_{01}	γ_{10}	γ_{12}	α
$\frac{14.0}{9.0}$	$\frac{1.0}{9.0}$	$-\frac{17.0}{6.0}$	$\frac{3.0}{2.0}$	$\frac{3.0}{2.0}$	$-\frac{1.0}{6.0}$	$-\frac{5.0}{9.0}$	$-\frac{1.0}{2.0}$	1.0	$\frac{1.0}{18.0}$	3.0	$\frac{1.0}{6.0}$	$\frac{1.0}{2.0}$	$\frac{1.0}{3.0}$

Table 2.1 Example coefficients for first derivative sixth-order interior and fourth-order boundary scheme. Scheme represented in equations 2.56 through 2.59. This scheme has two boundary rows, which is the minimum number for this order of accuracy, but more may be added.

This Mathematica script allows us to calculate these schemes for first and second derivatives, and in principle, to any order of accuracy. By applying these to discrete function values, we can approximate the spatial derivative of a function. These schemes must be stable and accurate so that when taking many derivatives across a simulation, the final model will be accurate.

Chapter 3

Results

In order for the CFD derivative schemes developed in Chapter 2 to be used for binary black hole simulations, they need to be stable and accurate. We can show, through a number of methods, that we can construct CFD operators that are both accurate and stable. If enough operators can be found that fit these conditions, then they show promise when applied to the numerical relativity simulations.

3.1 Accuracy

As we are now able to generate derivative operators and filters quickly, we are able to test their accuracy compared to known derivative solutions. By applying the generated operators to discretized functions, then comparing them to the analytic derivatives, we can measure the error across the domain.

Figures 3.1 and 3.3 depict comparisons between explicit and CFD schemes when computing a derivative. We can see in both cases, that a standard compact scheme exhibits an improvement in accuracy of about an order of magnitude in the central region of the domain. Also, as seen on the farthest points to the left and right in Figure 3.1, the error on the boundaries is many orders of magnitude less than the explicit schemes. This reduction in error shows that CFDs have the possibility to achieve the increase in accuracy desired.

Figure 3.1 depicts more improvement in accuracy using the boundary closure scheme and optimization techniques outlined by Kim [11]. The error at the boundary closures is about an order of accuracy better than the standard CFD schemes. The interior points use the optimization to further drive error down, preceding below machine precision for the machine used to calculate the derivative.

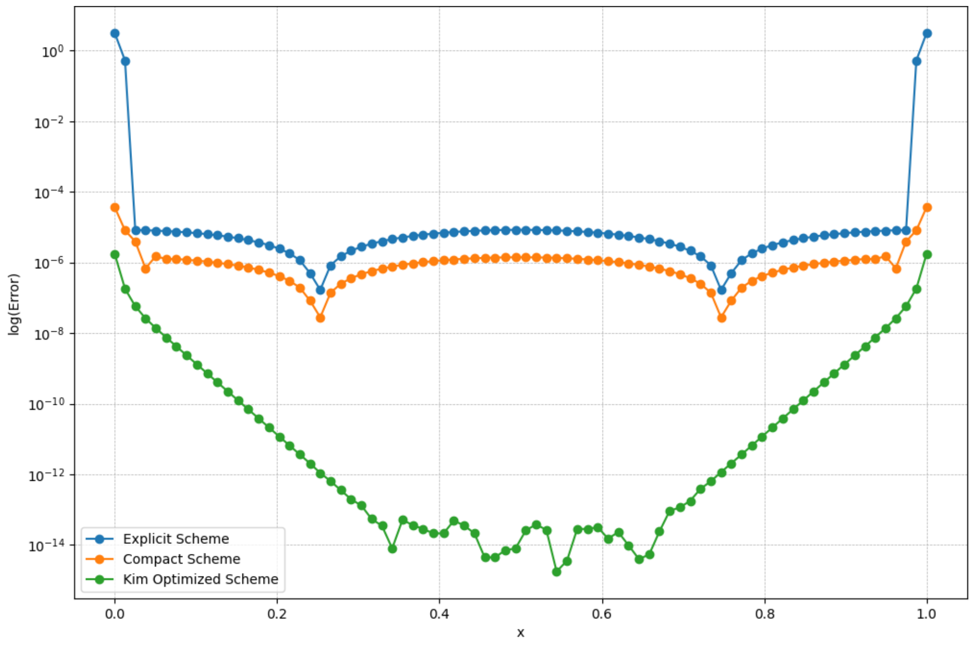

Figure 3.1 Fourth-order error comparison between derivative schemes. We took the derivative of $\sin(\pi x)$ using three differentiation schemes, then subtracted from the analytic derivative. This plot describes the difference between the analytic derivative and the numeric derivatives. The explicit scheme is a traditional way of taking derivatives, and clearly has a high amount of error on the boundaries. Using a standard compact scheme, the error on the boundaries falls by over four orders of magnitude, and around one order in the center. Finally, optimization techniques, such as those used by Kim [11] to generate the final operator, can decrease the error further without increasing the required number of function points.

3.2 Stability

It is important that the derivative schemes we use are also stable, not just accurate. We can use the solution to PDEs solved analytically to find the error over time created when solving the same equations with CFDs, providing a measure of stability for a given operator. Doing this for general relativity codes is difficult due to the complex and nonlinear nature of the Einstein equations. So we use simpler models when testing for stability, such as electromagnetic radiation.

Sharan, Brady and Livescu optimized their schemes for stability [13]. Figure 3.2 depicts the use of one of their CFD schemes compared to an explicit scheme currently in use. Each scheme depicted in the figure is stable, as the error quickly damps to a consistent value. The CFD schemes show slightly different behavior at larger times, but ultimately the oscillations in their error are smaller than those of the explicit schemes. More importantly, they maintain higher accuracy using the same resolution and remain stable.

We can see in Figure 3.3, the operators generated by Sharan, Brady and Livescu [13] are closer in accuracy to the standard Kim [11] scheme. Specifically, due to Dendro-GR having a high dependence on accurate boundaries, we hope to find a mix between the two schemes. The operators generated by Sharan, Brady and Livescu [13], though they are stable, do not have the accuracy improvement on the boundaries necessary for use in Dendro-GR.

3.2.1 Eigenvalue Spectra

One way of testing stability without performing long and costly simulations is to find the eigenvalues of the $P^{-1}Q$ matrix. This is called the eigenvalue spectra. Most of the eigenvalues are complex, and analyzing the real portions of these values provides a method of testing stability. As in Figure 3.4, if the real part of all the eigenvalues are negative, the CFD scheme is likely stable. This provides a relatively simple way to test the possible stability of these operators. Then they can be tested by running numerical simulations with known solutions.

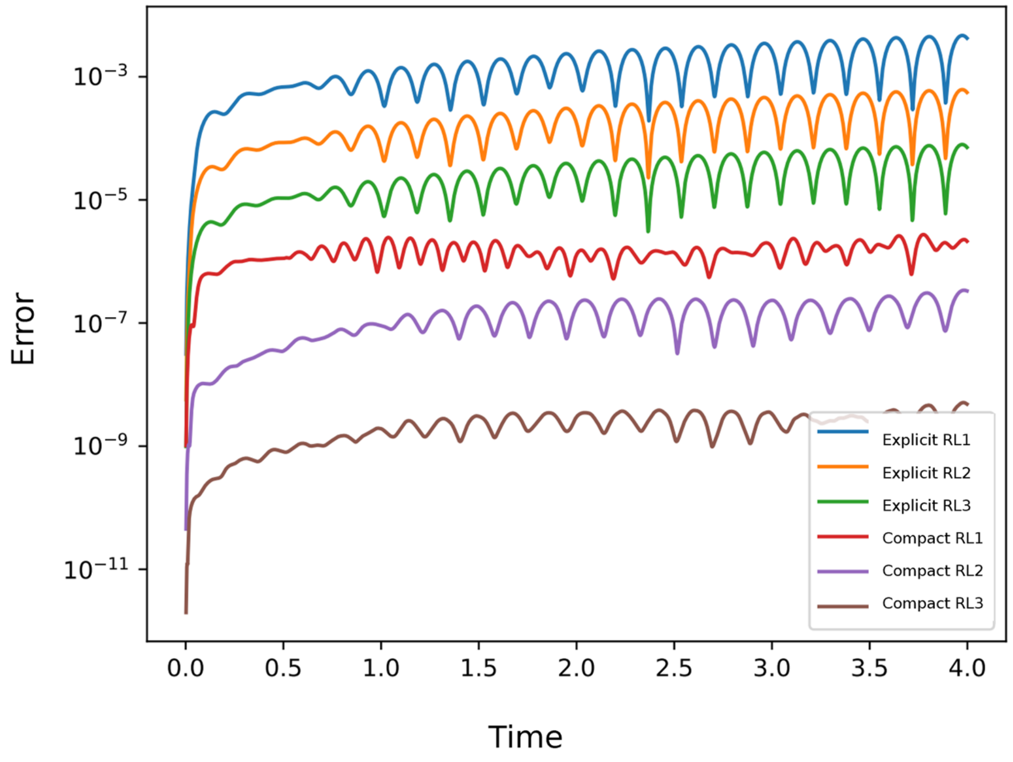

Figure 3.2 Error when evolving the 3D Maxwell equations. Both an explicit derivative scheme and a compact derivative scheme are used to approximate the solution of a standing wave in a box using Maxwell's equations, then subtracted from the analytic solution to find the error. The CFDs were generated by Nek, Brady, and Livescu [13]. Each scheme is sixth order accurate. Explicit scheme 1 and compact scheme 1 use the same number of grid points to approximate the solution, each of the following schemes increase in points together. As the error on none of the curves diverges exponentially, we can conclude that they are all stable for this equation. The error in the compact schemes are consistently more than two orders of magnitude lower than the explicit scheme with the same number of points. This shows promise that CFDs will provide increased accuracy when solving other PDEs, without being more unstable than the explicit schemes.

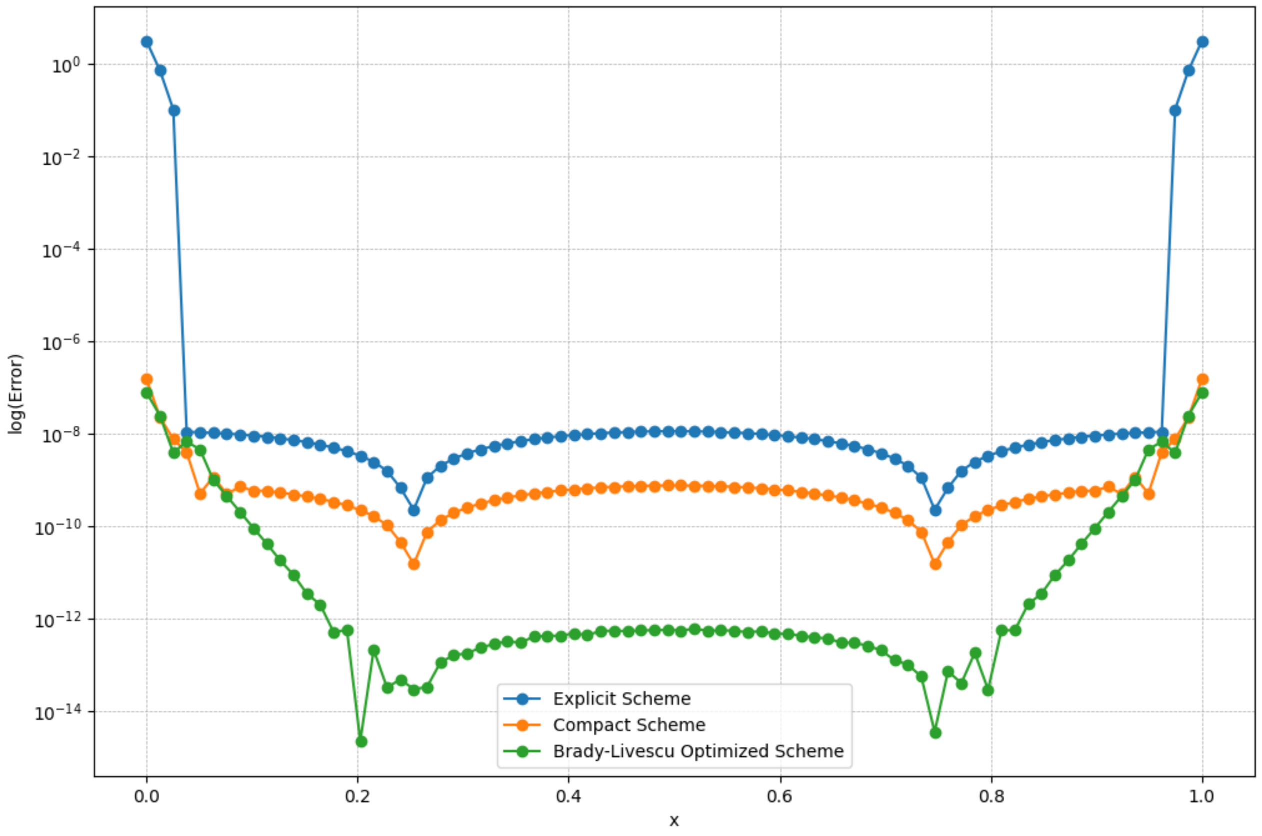

Figure 3.3 Sixth-order error comparison between different schemes. The derivative of $\sin(\pi x)$ was taken using three differentiation schemes, then subtracted from the analytic derivative. This plot describes the difference between the analytic derivative and the numeric derivatives. The explicit scheme is a traditional way of taking derivatives, and clearly has a high amount of error on the boundaries. Using a standard compact scheme, the error on the boundaries falls by over four orders of magnitude, and around one order in the center. Using optimization techniques, Nek, Brady, and Livescu [13] were able to increase the stability of their operator. This optimization sacrificed accuracy on the boundaries for greater stability. Combining their methods with other methods to increase accuracy might allow for overall improvements in the CFD schemes.

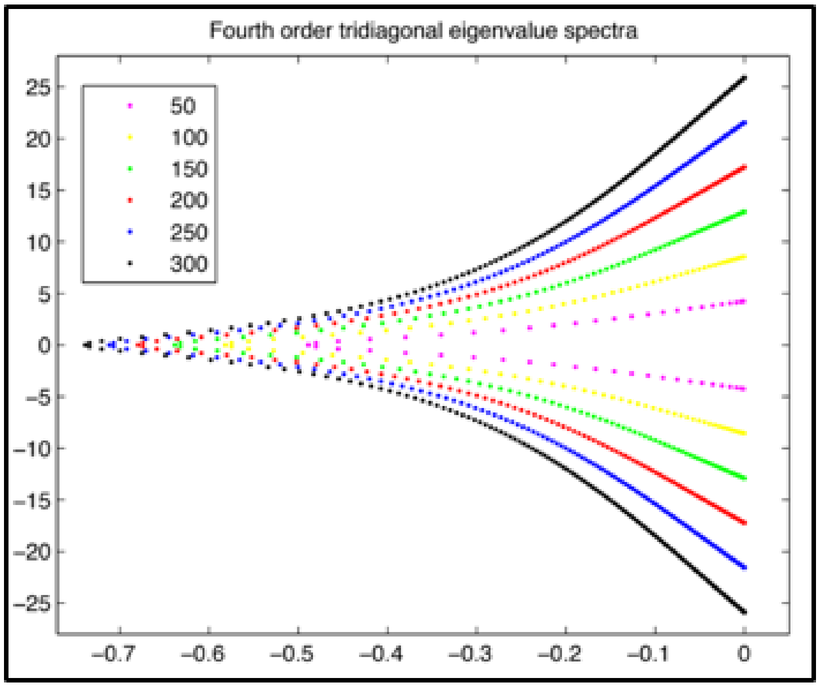

Figure 3.4 Example Eigenvalue Spectra from Tyler [9]. This figure depicts the eigenvalue spectra of $P^{-1}Q$ for the same scheme of different domain sizes N. As N represents the number of points across the grid, it must be positive and real. The resolution of the grid across which the derivative is approximated increases as N increases. Stability of an operator is likely if the real part of each eigenvalue is negative. The imaginary part of each eigenvalue is does not provide any information about stability, but correctly calculated eigenvalue spectra are symmetrical across the real-axis.

3.3 CFDs and Filtering Solving Maxwell's Equations

We have applied the filtering schemes discussed in Section 2.3 to first derivative schemes we generated, and used them to solve the 3D Maxwell equations using Dendro-GR. Specifically, we solved for a propagating magnetic dipole. Using a Wavelet Adaptive Multi-Resolution grid with boundaries at ± 20 in all three directions, a wavelet tolerance of 10^{-5} and a grid spacing of 0.2083333 in the highest resolution areas. This gave us a test bed within which to compare filtered and non-filtered CFDs to explicit schemes. As illustrated in Figure 3.5, we have shown that filtering the operators has the potential to consistently decrease the L2 error across multiple fields in a system of coupled partial differential equations.

In Figure 3.5, there are three figures each examining the error from solving the three dimensional Maxwell's equations using different schemes. This error was found by subtracting our solution from the exact analytic solution. The three plots are for the error in the x component of the magnetic and electric fields, and for the divergence of the magnetic field. In each case we see a similar reduction of error between the unfiltered CFD schemes and the filtered CFD schemes.

It is clear that the measure of error reduction changes depending on which CFD scheme is used together with the particular filtering scheme. This likely means that filtering operators and CFD operators can be optimized in tandem to maximize their combined efforts for both stability, and error reduction. This is work that needs to be explored in the future.

The top left figure in Figure 3.5 shows the L2 error in the magnetic field throughout the simulation. We can see that the KIMBYU_1, a fourth-order scheme developed at BYU, has lower error across much of the simulation than both the explicit fourth-order (E4) and explicit sixth-order (E6) schemes. We can see within differently optimized CFDs difference in the error amounts. Note that KIMBYU_4, without filtering, has a larger error content than KIMBYU_1. When applying filtering to both of the CFD scheme, we note that the the error drops significantly. In particular, the KIMBYU_4 scheme operates at the level of the KIMBYU_1 both filtered and unfiltered. Both Filter_KIMBYU_1 and Filter_KIMBYU_4 use a sixth-order tridiagonal filter with α_f being 0.4.

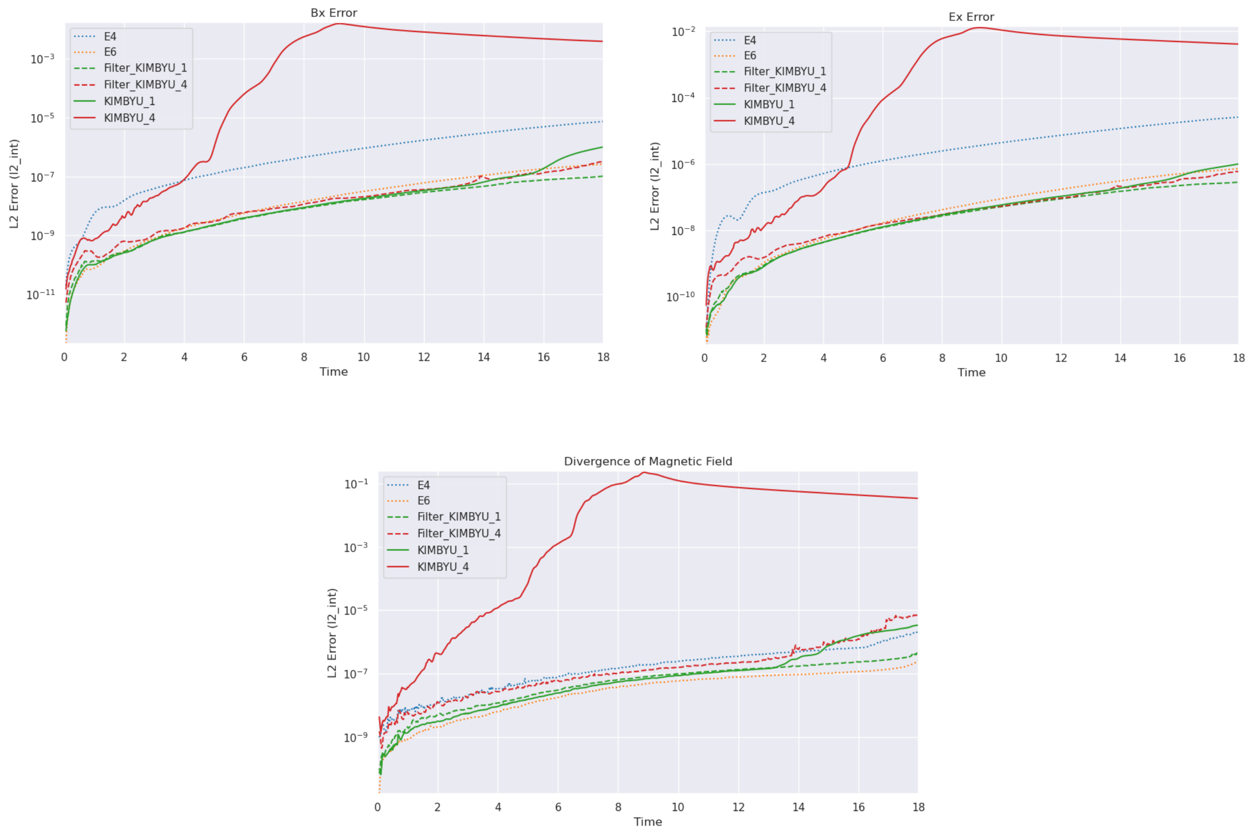

Figure 3.5 Comparison of Error Across Filtered and Non-Filtered Schemes using 3D Maxwell Equations. These plots represent the L2 error of multiple solves of the 3D Maxwell Equation using different operators. The blue and orange dotted curves represent the error when using a fourth-order (E4) and a sixth-order (E6) explicit scheme respectively. The red and green curves represent the error when using two different optimized fourth-order CFD schemes we generated at BYU (KIMBYU_1, KIMBYU_4). The red and green dashed lines represent these same operators, with filtering applied (Filter_KIMBYU_1, Filter_KIMBYU_4). The filter was a sixth-order tridiagonal form of the filters discussed in Section 2.3, with an α_f value of 0.4.

The top right figure depicts the L2 error in the electric field, and the bottom filter shows the L2 error in the divergence of the magnetic field. The same effects observed in the error in the magnetic field are mostly also observed in the other fields. This suggests that the individual operators do not interact with the different fields in wildly different ways. The effects in the three plots shown are mirrored in all the other fields that are evolved.

Overall, the figure shows that filtering can have an appreciable effect on lowering the error when using CFD schemes to solve partial differential equations. KIMBYU_4, the scheme with high error, and KIMBYU_1 both had their L2 error lowered. In this case, KIMBYU_4 had its error lowered much more, suggesting that specific filters can have much larger effects when paired with certain operators.

Chapter 4

Discussion and Conclusions

Binary black hole mergers are the sources for most of the observable gravitational waves in the universe. They can thus be used to probe deep into the past of our universe. Studying these objects requires in-depth models to generate waveforms to compare against observations. These models can struggle to encapsulate the highest energy and largest mass ratio systems [4]. We hope that by using CFDs, the computational costs may be lowered enough to explore these systems, without significant accuracy loss. This requires us to find stable, accurate schemes to be used.

Using new compact schemes, rather than existing explicit schemes, we have calculated families of CFDs that can be used to approximate derivatives. We have used the Taylor series expansion at each point to calculate these coefficients and ensure each operator correctly and accurately approximates solutions to PDEs. We have created a Mathematica script to automate this process and provide a starting place for further work improving accuracy and stability.

As shown in Section 3, we have found evidence that CFDs can be stable and accurate over long time periods while approximating solutions to nonlinear differential equations. We hope we can extend these test cases to more complicated problems, and eventually to black hole mergers. However, more work must be done to explore a larger variety of stable second derivative operators to prepare for these more complex test cases.

Overall, we have shown that CFDs are more accurate than excplicit schemes across the entire domain, but especially on the boundary. This increase in accuracy does not correspond to an increase in required function points, therefore allowing similar computational time for an increased accuracy. We have both first and second derivative schemes that have shown this increased accuracy, and with optimization techniques we will be able to further drive down error.

Current work only includes stable tridiagonal second derivatives with simple boundary closure schemes. More work will need to be done to explore other stable operators due to Dendro-GR's requirements for multiple boundaries across the entire grid. The results we have achieved provide evidence that these more complex schemes exist and will likely continue being more accurate than explicit differentiation schemes.

In the future we hope to test techniques of filtering and test new operators to check for stability. As we have developed multiple numeric models that use CFDs, we are able to expedite this process and continue working to find these operators. Though more work needs to be done, these operators present a chance to significantly improve the capabilities of current binary black hole simulations.

Compact finite differencing is a method of estimating derivatives that has been used in engineering applications but has not been explored in the domain of physics up until this point. We hope that using CFDs will increase the performance of current binary black hole simulations. This comes from an increase in accuracy without an increase in computational cost, allowing for more complex systems to be analyzed. Future implementations of this strategy promise better gravitational waveforms for comparison with data gathered from next generation gravitational wave detectors.

Appendix A

Analytic Matrix Inverses

One way to further decrease the computational cost of compact finite differencing is to improve algorithms for taking the inverse of the left hand side of the schemes. Due to the mostly uniform structure of each of these matrices, in early work on CFDs, we attempted to find analytic methods of taking the inverse of each of these. This involves finding the determinants of the matrices, then their cofactor expansions. We have also reparameterized these determinants to make them easier to work with. Our work reviewing this approach is reported below.

A.1 No Boundary Condition

First we consider a tridiagonal scheme with no set boundary conditions. Consider the following $k \times k$ tridiagonal matrix:

$$M = \begin{pmatrix} D & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & D & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & D & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & D \end{pmatrix} \tag{A.1}$$

where D is a constant on the main diagonal and $k \geq 1$.

The term M_k is the determinant of the basic tridiagonal matrix of order k, given by:

$$M_k = \frac{1}{\eta} \frac{1}{2^{k+1}} \left[(D+\eta)^{k+1} - (D-\eta)^{k+1} \right] \tag{A.2}$$

where $\eta = \sqrt{D^2 - 4}$.

Reparameterizing D in this case, we find that:

$$D = \begin{cases} -2\cosh\lambda & \text{if } D \leq -2 \\ 2\cos\lambda & \text{if } |D| < 2 \\ 2\cosh\lambda & \text{if } D \geq 2 \end{cases} \tag{A.3}$$

Plugging this into our determinant M_k we find the following:

$$M_k = \begin{cases} \frac{(-1)^k \sinh\left((k+1)\lambda\right)}{\sinh\lambda} & \text{if } D \leq -2 \\ \frac{\sin\left((k+1)\lambda\right)}{\sin\lambda} & \text{if } |D| < 2 \\ \frac{\sinh\left((k+1)\lambda\right)}{\sinh\lambda} & \text{if } D \geq 2 \end{cases} \tag{A.4}$$

We now put these pieces together and explore the cofactors of M. We can find several patterns in its cofactor matrix, c. Using these patterns, we can find that the cofactor matrix is as follows:

$$c_{ij} = (-1)^{i+j} M_{i-1} M_{k-j} \quad i \leq j \tag{A.5}$$

The cofactor matrix is symmetric across the diagonal, so this equation can still be used to solve for the entire matrix not just the top half. But, since we now have they cofactor matrix equation, we can use it and the determinant to find the analytic inverse of M, which is as follows:

$$(M^{-1})_{ij} = \begin{cases} -\frac{\cosh[(k+1-|j-i|\lambda)]-\cosh[(k+1-j-1)\lambda]}{2\sinh\lambda\,\sinh[(k+1)\lambda]} & \text{if } D \leq -2 \\ (-1)^{i+j+1}\frac{\cos[(k+1-|j-i|)\lambda]-\cos[(k+1-j-i)\lambda]}{2\sin\lambda\,\sin[(k+1)\lambda]} & \text{if } |D| < 2 \\ (-1)^{i+j}\frac{\cosh[(k+1-|j-i|\lambda)]-\cosh[(k+1-j-i)\lambda]}{2\sinh\lambda\,\sinh[(k+1)\lambda]} & \text{if } D \geq 2 \end{cases} \tag{A.6}$$

A.2 Schemes with Two Boundaries

We start by exploring a tridiagonal scheme with a single boundary row on each side. Consider the following $k \times k$ matrix for a compact finite difference scheme on the boundaries:

$$\hat{M} = \begin{pmatrix} D & \beta & \gamma & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & D & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & D & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \gamma & \beta & D \end{pmatrix} \tag{A.7}$$

where D is a constant on the main diagonal, β and γ are parameters found by solving the scheme to the desired order, and $k \geq 3$.

We will refer to the determinant of this matrix as $\hat{M}_k$. We have found that that $\hat{M}_k$ satisfies the following recursion relation:

$$\hat{M}_k = D^2 M_{k-2} - 2\beta D M_{k-3} + (2\gamma D + \beta^2) M_{k-4} - 2\gamma\beta M_{k-5} + \gamma^2 M_{k-6} \tag{A.8}$$

Through some simplifying, we find $\hat{M}_k$ to be as follows:

$$\hat{M}_k = \frac{1}{\eta}\frac{1}{2^{k-1}}\Bigg\{D^2\left[(D+\eta)^{k-1} - (D-\eta)^{k-1}\right] \tag{A.9}$$

$$-4\beta D\left[(D+\eta)^{k-2} - (D-\eta)^{k-2}\right] \tag{A.10}$$

$$+4(\beta^2 + 2\gamma D)\left[(D+\eta)^{k-3} - (D-\eta)^{k-3}\right] \tag{A.11}$$

$$-16\gamma\beta\left[(D+\eta)^{k-4} - (D-\eta)^{k-4}\right] \tag{A.12}$$

$$+16\gamma^2\left[(D+\eta)^{k-5} - (D-\eta)^{k-5}\right]\Bigg\} \tag{A.13}$$

where, again, $\eta = \sqrt{D^2 - 4}$.

Reparameterizing this case, we find that:

$$\hat{M}_k = \begin{cases} \frac{1}{\sin\lambda}\Big[4\cos^2\lambda\sin((k-1)\lambda) - 4\beta\cos\lambda\sin((k-2)\lambda) & \\ +(\beta^2 + 4\gamma\cos\lambda)\sin((k-3)\lambda) - 2\gamma\beta\sin((k-4)\lambda) & \text{if } \eta^2 < 0 \\ +\gamma^2\sin((k-5)\lambda)\Big] & \\ \frac{1}{\sinh\lambda}\Big[4\cosh^2\lambda\sinh((k-1)\lambda) - 4\beta\cosh\lambda\sinh((k-2)\lambda) & \\ +(\beta^2 + 4\gamma\cosh\lambda)\sinh((k-3)\lambda) - 2\gamma\beta\sinh((k-4)\lambda) & \text{if } \eta^2 \geq 0 \\ +\gamma^2\sinh((k-5)\lambda)\Big] & \end{cases} \tag{A.14}$$

We now put these pieces together and explore the cofactors of $\hat{M}$. We can find several patterns in its cofactor matrix, $\hat{C}$. Consider the following matrix:

$$\hat{C} = \begin{pmatrix} x & x & x & x & \cdots & x & x & y & z \\ w & x & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & y & z \\ & \vdots & & & \ddots & & & \vdots & \\ z & y & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & x & w \\ z & y & x & x & \cdots & x & x & x & x \end{pmatrix} \tag{A.15}$$

where,

$$x = (-1)^{i+j}\hat{T}_{i-1-\max(0,i-j)}\hat{T}_{k-j-\max(0,i-j)} \tag{A.16}$$

$$y = (-1)^{i+j}(D-\gamma)\hat{T}_{k-i} \tag{A.17}$$

$$z = (-1)^{i+j}(\beta - D\gamma)\hat{T}_{k-i} \tag{A.18}$$

$$w = (-1)^{i+j}(\beta\hat{T}_{k-2} - \gamma\hat{T}_{k-3}) \tag{A.19}$$

if $j \le \lceil k/2 \rceil$, and

$$x = (-1)^{i+j}\hat{T}_{k-i-\max(0,j-i)}\hat{T}_{j-1-\max(0,j-i)} \tag{A.20}$$

$$y = (-1)^{i+j}(D-\gamma)\hat{T}_{i-1} \tag{A.21}$$

$$z = (-1)^{i+j}(\beta - D\gamma)\hat{T}_{i-1} \tag{A.22}$$

$$w = (-1)^{i+j}(\beta\hat{T}_{k-2} - \gamma\hat{T}_{k-3}) \tag{A.23}$$

if $j > \lceil k/2 \rceil$. Within these equations, the $\hat{T}_k$ terms represent determinants of arbitrarily sized matrices explored in Section A.3.

For computational purposes, it is more helpful to only compute the cofactors for $j \leq \lceil k/2 \rceil$. Due to the matrix's 180-degree rotational symmetry, the cofactors for $j > \lceil k/2 \rceil$ can be found using the relation

$$\hat{C}_{i,j} = \hat{C}_{k+1-i,k+1-j}. \tag{A.24}$$

Now we explore an approach similar to Mehra et al. [14], with the following form of the $k \times k$ tridiagonal matrix to represent boundary conditions:

$$\overline{M} = \begin{pmatrix} D & \alpha & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \beta & D & \beta & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta & D & \beta \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha & D \end{pmatrix} \tag{A.25}$$

where D is a constant on the main diagonal, α and β are parameters found by solving the scheme to the desired order, and $k \geq 4$.

We will refer to the determinant of this matrix as $\overline{M}_k$. We have found that $\overline{M}_k$ satisfies the following recursion relation:

$$\overline{M}_k = (D^4 - 2\alpha\beta D^2 + \alpha^2\beta^2)M_{k-4} + 2(\alpha\beta^2 D - \beta D^3)M_{k-5} + (\beta^2 D^2)M_{k-6} \tag{A.26}$$

Through some simplifying, we can express $\overline{M}_k$ as follows:

$$\overline{M}_k = \frac{1}{\eta}\frac{1}{2^{k-3}}\Big\{(D^2-\alpha\beta)^2\left[(D+\eta)^{k-3}-(D-\eta)^{k-3}\right] \tag{A.27}$$

$$+4D\beta(\alpha\beta-D^2)\left[(D+\eta)^{k-4}-(D-\eta)^{k-4}\right] \tag{A.28}$$

$$+4D^2\beta^2\left[(D+\eta)^{k-5}-(D-\eta)^{k-5}\right]\Big\} \tag{A.29}$$

where, again, $\eta = \sqrt{D^2-4}$.

Reparameterizing this new case, we find that:

$$\overline{M}_k = \begin{cases} \frac{1}{\sin\lambda}\Big[(4\cos^2\lambda-\alpha\beta)^2\sin((k-3)\lambda) & \\ +4\beta\cos\lambda(\alpha\beta-4\cos^2\lambda))\sin((k-4)\lambda) & \text{if } \eta^2<0 \\ +4\beta^2\cos^2\lambda\sin((k-5)\lambda)\Big] & \\ \frac{1}{\sin\lambda}\Big[(4\cosh^2\lambda-\alpha\beta)^2\sinh((k-3)\lambda) & \\ +4\beta\cosh\lambda(\alpha\beta-4\cosh^2\lambda))\sinh((k-4)\lambda) & \text{if } \eta^2\geq 0 \\ +4\beta^2\cosh^2\lambda\sinh((k-5)\lambda)\Big] & \end{cases} \tag{A.30}$$

Now we consider the cofactor expansion of this scheme. We can find several patterns in its cofactor matrix. Consider the following matrix:

$$\overline{C} = \begin{pmatrix} u & v & v & v & \cdots & v' & v' & y' & z' \\ w & x & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & y & z \\ & \vdots & & & \ddots & & & \vdots & \\ z & y & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & x & w \\ z' & y' & v' & v' & \cdots & v & v & v & u \end{pmatrix} \tag{A.31}$$

where,

$$x = (-1)^{i+j}\overline{T}_{i-1-\max(0,i-j)}\overline{T}_{k-j-\max(0,i-j)} \tag{A.32}$$

$$y = (-1)^{i+j}D\beta\overline{T}_{k-i} \tag{A.33}$$

$$z = (-1)^{i+j}\alpha\beta\overline{T}_{k-i} \tag{A.34}$$

$$u = (-1)^{i+j}(D\overline{T}_{k-2} - \beta\overline{T}_{k-3}) \tag{A.35}$$

$$v = (-1)^{i+j}\beta\overline{T}_{k-j} \tag{A.36}$$

$$w = (-1)^{i+j}\alpha\overline{T}_{k-i} \tag{A.37}$$

$$v' = (-1)^{i+j}\beta\overline{M}_{j-1} \tag{A.38}$$

$$y' = (-1)^{i+j}D\beta^2 \tag{A.39}$$

$$z' = (-1)^{i+j}\alpha\beta^2 \tag{A.40}$$

$$\tag{A.41}$$

if $j \leq \lceil k/2 \rceil$. The $\overline{T}_k$ terms represent determinants of matrices explored in Section A.3.

It is worth noting that in the top row of the matrix, v continues on until $j = \lceil k/2 \rceil$, and then $j = \lceil k/2 \rceil + 1$ is v'. A similar transition occurs on the bottom row of the matrix.

Similar to the previous case, for computational purposes, it is more helpful to only compute the cofactors for $j \leq \lceil k/2 \rceil$. Due to the matrix's 180-degree rotational symmetry, the cofactors for $j > \lceil k/2 \rceil$ can be found using the relation

$$\overline{C}_{i,j} = \overline{C}_{k+1-i,k+1-j} \tag{A.42}$$

Finally, taking an approach similar to Tyler [9], we have the following form of the $k \times k$ tridiagonal matrix to represent boundary conditions:

$$\acute{M} = \begin{pmatrix} D & \alpha & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \beta & D & \gamma & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \gamma & D & \beta \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha & D \end{pmatrix} \tag{A.43}$$

where D is a constant on the main diagonal, α, β, and γ are parameters found by solving the scheme to the desired order, and $k \geq 4$.

We will refer to the determinant of this matrix as $\acute{M}_k$. We find that this determinant, $\acute{M}_k$, satisfies the following recursion relation:

$$\acute{M}_k = (D^4 - 2\alpha\beta D^2 + \alpha^2\beta^2)M_{k-4} + 2(\alpha\beta\gamma D - \gamma D^3)M_{k-5} + (\gamma^2 D^2)M_{k-6} \tag{A.44}$$

Through some simplifying, we can express $\acute{M}_k$ as follows:

$$\acute{M}_k = \frac{1}{\eta}\frac{1}{2^{k-3}}\Bigg\{(D^2 - \alpha\beta)^2\Big[(D+\eta)^{k-3} - (D-\eta)^{k-3}\Big] \tag{A.45}$$

$$+4(D\alpha\beta\gamma - \gamma D^3)\Big[(D+\eta)^{k-4} - (D-\eta)^{k-4}\Big] \tag{A.46}$$

$$+4D^2\gamma^2\Big[(D+\eta)^{k-5} - (D-\eta)^{k-5}\Big]\Bigg\} \tag{A.47}$$

where, again, $\eta = \sqrt{D^2 - 4}$.

Reparameterizing $\acute{M}_k$, we find that:

$$\acute{M}_k = \begin{cases} \frac{1}{\sin\lambda}\Big[(4\cos^2\lambda - \alpha\beta)^2 \sin((k-3)\lambda) & \\ +4\gamma\cos\lambda(\alpha\beta - 4\cos^2\lambda))\sin((k-4)\lambda) & \text{if } \eta^2 < 0 \\ +4\gamma^2\cos^2\lambda\sin((k-5)\lambda)\Big] & \\ \frac{1}{\sin\lambda}\Big[(4\cosh^2\lambda - \alpha\beta)^2 \sinh((k-3)\lambda) & \\ +4\gamma\cosh\lambda(\alpha\beta - 4\cosh^2\lambda))\sinh((k-4)\lambda) & \text{if } \eta^2 \geq 0 \\ +4\gamma^2\cosh^2\lambda\sinh((k-5)\lambda)\Big] & \end{cases} \tag{A.48}$$

We explore cofactor expansion of this scheme below. We can find several patterns in its cofactor matrix. Consider the following matrix:

$$\acute{C} = \begin{pmatrix} u & v & v & v & \cdots & v' & v' & y' & z' \\ w & x & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & y & z \\ & \vdots & & & \ddots & & & \vdots & \\ z & y & x & x & \cdots & x & x & y & z \\ z & y & x & x & \cdots & x & x & x & w \\ z' & y' & v' & v' & \cdots & v & v & v & u \end{pmatrix} \tag{A.49}$$

where,

$$x = (-1)^{i+j}\acute{T}_{i-1-\max(0,i-j)}\acute{T}_{k-j-\max(0,i-j)} \tag{A.50}$$

$$y = (-1)^{i+j}D\gamma\acute{T}_{k-i} \tag{A.51}$$

$$z = (-1)^{i+j}\alpha\gamma\acute{T}_{k-i} \tag{A.52}$$

$$u = (-1)^{i+j}(D\acute{T}_{k-2} - \gamma\acute{T}_{k-3}) \tag{A.53}$$

$$v = (-1)^{i+j}\beta\acute{T}_{k-j} \tag{A.54}$$

$$w = (-1)^{i+j}\alpha\acute{T}_{k-i} \tag{A.55}$$

$$v' = (-1)^{i+j}\beta\acute{M}_{j-1} \tag{A.56}$$

$$y' = (-1)^{i+j}D\beta\gamma \tag{A.57}$$

$$z' = (-1)^{i+j}\alpha\beta\gamma \tag{A.58}$$

$$\tag{A.59}$$

if $j \leq \lceil k/2 \rceil$. Where the $\acute{T}_k$ terms are determinants of general matrices explored in Section A.3.

It is worth noting that in the top row of the matrix, v continues on until $j = \lceil k/2 \rceil$, and then $j = \lceil k/2 \rceil + 1$ is v'. A similar transition occurs on the bottom row of the matrix.

Similar to the previous case, for computational purposes, it is more helpful to only compute the cofactors for $j \leq \lceil k/2 \rceil$. Due to the matrix's 180-degree rotational symmetry, the cofactors for $j > \lceil k/2 \rceil$ can be found using the relation

$$\acute{C}_{i,j} = \acute{C}_{k+1-i,k+1-j}. \tag{A.60}$$

A.3 Determinants of Partial Matrices

Consider now the following $k \times k$ tridiagonal matrix for a compact finite difference scheme on the boundaries:

$$\hat{T} = \begin{pmatrix} D & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & D & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & D & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \gamma & \beta & D \end{pmatrix} \tag{A.61}$$

where D is a constant on the main diagonal, and β and γ are parameters found by solving the scheme to the desired order.

We will refer to the determinant of this matrix as $\hat{T}_k$. Similarly to the $\hat{M}$ case, we can show that $\hat{T}_k$ satisfies the following recursion relation:

$$\hat{T}_k = DM_{k-1} - \beta M_{k-2} + \gamma M_{k-3} \tag{A.62}$$

Through some simplifying, we can express $\hat{T}_k$ as follows:

$$\hat{T}_k = \frac{1}{\eta}\frac{1}{2^k}\left\{ D\left[(D+\eta)^k - (D-\eta)^k\right] \right. \tag{A.63}$$

$$-2\beta\left[(D+\eta)^{k-1} - (D-\eta)^{k-1}\right] \tag{A.64}$$

$$\left. +4\gamma\left[(D+\eta)^{k-2} - (D-\eta)^{k-2}\right] \right\} \tag{A.65}$$

where $\eta = \sqrt{D^2-4}$.

Consider the following $k \times k$ tridiagonal matrix for a compact finite difference scheme on the boundaries:

$$\overline{T} = \begin{pmatrix} D & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & D & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta & D & \beta \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha & D \end{pmatrix} \tag{A.66}$$

where D is a constant on the main diagonal, and α and β are parameters found by solving the scheme to the desired order.

We will refer to the determinant of this matrix as $\overline{T}_k$. We find that $\overline{T}_k$ satisfies the following recursion relation:

$$\overline{T}_k = (D^2 - \alpha\beta)M_{k-2} - \beta D M_{k-3} \tag{A.67}$$

Through some simplifying, we can express $\overline{T}_k$ as follows:

$$\overline{T}_k = \frac{1}{\eta}\frac{1}{2^{k-1}}\Big\{(D^2 - \alpha\beta)\Big[(D+\eta)^{k-1} - (D-\eta)^{k-1}\Big] \tag{A.68}$$

$$-2\beta D\Big[(D+\eta)^{k-2} - (D-\eta)^{k-2}\Big]\Big\} \tag{A.69}$$

where, again, $\eta = \sqrt{D^2 - 4}$.

Lastly, we consider the following $k \times k$ tridiagonal matrix for a compact finite difference scheme on the boundaries:

$$\acute{T} = \begin{pmatrix} D & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & D & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & D & 1 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & D & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \gamma & D & \beta \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha & D \end{pmatrix} \tag{A.70}$$

where D is a constant on the main diagonal, and α, β, and γ are parameters found by solving the scheme to the desired order.

We will refer to the determinant of this matrix as $\acute{T}_k$. We find that $\acute{T}_k$ satisfies the following recursion relation:

$$\acute{T}_k = (D^2 - \alpha\beta)M_{k-2} - \gamma D M_{k-3} \tag{A.71}$$

Through some simplifying, we can express $\acute{T}_k$ as follows:

$$\acute{T}_k = \frac{1}{\eta}\frac{1}{2^{k-1}}\Big\{(D^2 - \alpha\beta)\Big[(D+\eta)^{k-1} - (D-\eta)^{k-1}\Big] \tag{A.72}$$

$$-2\gamma D\Big[(D+\eta)^{k-2} - (D-\eta)^{k-2}\Big]\Big\} \tag{A.73}$$

where, again, $\eta = \sqrt{D^2 - 4}$.

Bibliography

[1] A. Dirkes, “Gravitational waves — A review on the theoretical foundations of gravitational radiation,” International Journal of Modern Physics A **33,** 1830013 (2018).

[2] T. eLISA Consortium *et al.*, “The Gravitational Universe,”, 2013.

[3] F. Pretorius, “Evolution of Binary Black-Hole Spacetimes,” Physical Review Letters 95 (2005).

[4] J. Healy and C. O. Lousto, “Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits,” Physical Review D 105 (2022).

[5] T. Regimbau, M. Evans, N. Christensen, E. Katsavounidis, B. Sathyaprakash, and S. Vitale, “Digging Deeper: Observing Primordial Gravitational Waves below the Binary-Black-Hole-Produced Stochastic Background,” Physical Review Letters 118 (2017).

[6] D. Ferguson, K. Jani, P. Laguna, and D. Shoemaker, “Assessing the readiness of numerical relativity for LISA and 3G detectors,” Physical Review D 104 (2021).

[7] M. Fernando, D. Neilsen, Y. Zlochower, E. W. Hirschmann, and H. Sundar, “Massively parallel simulations of binary black holes with Dendro-GR,”, 2022.

[8] M. Ishii, M. Fernando, K. Saurabh, B. Khara, B. Ganapathysubramanian, and H. Sundar, “Solving PDEs in space-time: 4D tree-based adaptivity, mesh-free and matrix-free approaches,” In *Proceedings of the International Conference for High Performance Computing, Networking,*

Storage and Analysis, SC '19 (Association for Computing Machinery, New York, NY, USA, 2019).

[9] J. G. Tyler, Master's thesis, Brigham Young University, 2007.

[10] X. Zhao and C. Scalo, "A Compact-Finite-Difference-Based Numerical Framework for Adaptive-Grid-Refinement Simulations of Vortex-Dominated Flows," In , (2020).

[11] J. W. Kim, "Optimised boundary compact finite difference schemes for computational aeroacoustics," Journal of Computational Physics **225,** 995–1019 (2007).

[12] P. Brady and D. Livescu, "High-order, stable, and conservative boundary schemes for central and compact finite differences," Computers Fluids **183,** 84–101 (2019).

[13] N. Sharan, P. T. Brady, and D. Livescu, "Time Stability of Strong Boundary Conditions in Finite-Difference Schemes for Hyperbolic Systems," SIAM Journal on Numerical Analysis **60,** 1331–1362 (2022).

[14] M. Mehra and K. S. Patel, "Algorithm 986: A Suite of Compact Finite Difference Schemes," ACM Trans. Math. Softw. 44 (2017).

Index