6-11-2022

From Equal-mass to Extreme-mass-ratio Binary Inspirals: Simulation Tools for Next Generation Gravitational Wave Detectors

Samuel Douglas Cupp
Louisiana State University and Agricultural and Mechanical College

FROM EQUAL-MASS TO EXTREME-MASS-RATIO BINARY INSPIRALS: SIMULATION TOOLS FOR NEXT GENERATION GRAVITATIONAL WAVE DETECTORS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Department of Physics and Astronomy

by
Samuel Douglas Cupp
B.S. in Physics, Austin Peay State University, 2015
August 2022

Acknowledgments

While we all like to imagine that much of our accomplishments are the results of our own efforts, the truth is that much of what we do relies on others investing in our growth and progress. Science without collaboration will inevitably fall short of what could have been done, and students without mentors will learn far less. I, of course, am no different, and many people have contributed to my success.

I thank my committee members for the time they spent reviewing my thesis and listening to my general exam and thesis defense. I particularly thank Peter Diener and Steve Brandt as my advisor and unofficial second advisor, respectively. They both put in a lot of work to help me with research and general career advice/advancement. The office staff also deserve thanks for the behind-the-scenes work. Paige Whittington particularly had to suffer my many emails regarding graduation deadlines and scheduling, so thanks for bearing with me.

Several collaborators also worked with me heavily during this research. Roland Haas contributed to my work on Carpet and CarpetX and helped me with understanding the unpleasant guts inside Cactus. Barry Wardell, Niels Warburton, and Adam Pound spent a lot of time at the Capra meetings and through email correspondence to help with the Lorenz gauge code. Barry and Niels also wrote the effective source code I use for the RWZ gauge code. I also have to thank Leor Barack for taking the time to dig up errata for a paper from 15 years ago to help me figure out what I needed to do to reproduce the derived equations in that paper.

The largest contributors are certainly my parents, Sam and Melissa Cupp, who supported my academic journey and provided a stable environment for my growth and development as a person. Be it telling me to apply to REUs, helping me to move to Louisiana, or just providing a safety net to make student life easier, they put great effort into helping me succeed.

I also have to thank the faculty and staff in the Physics & Astronomy Department at Austin Peay State University for helping me immensely in my studies. The close-knit

community in the department helped most of us through the degree path, even when some semesters for my class went completely off-the-rails. Particularly, I thank B. Alex King and Justin Oelgoetz. I first met Alex during the Governor's School for Computational Physics, which I attended as a high schooler. From there all the way through my undergraduate degree he went way above the call of duty to help people in the department. I also appreciate how easy it was to convince him to teach classes for just me, though I think he might not see it the same way. On the research side, Justin advised my undergraduate research and helped me develop my skills as a computational scientist, which has benefited me greatly. Finally, the department secretary Sherry Bagwell deserves tons of praise. Without her constant work, I'm sure the entire department would collapse within hours, so thank you for all you do to keep the gears turning.

While there are too many people to mention during my time at APSU, a few stand out. Of the upperclassmen, James, Chris, and Robert were great friends and also put in a lot of hours helping us with classwork and research. In my class, Travis Tanner, Justin, Mees, and Sam Wyatt all made things way more fun and made the rough times bearable. While he only showed up near the end, Christian spiced things up for our senior year. I also have the distinction of being one of the few members of my class that spent all my time in the department, so I avoided my work by helping underclassmen with theirs. Ricky, Travis Hodge, Thomas, and Dominic provided welcome reprieve from my workload by asking me to do something I already knew how to do, for which I am very thankful. Finally, I especially want to thank Donny, with whom I probably spent the most time in undergrad. Though he wasn't in the department, he was a great roommate and friend. I hope you're enjoying Taiwan and your marriage.

I eventually ended up following Robert to LSU and was myself followed by Thomas, so presumably a new APSU graduate will soon come to replace me in the department. In addition to old faces, I met many new friends at LSU. While I won't try to thank or mention everyone, I will mention David and Nick. I spent many weekends playing board games in the

Armenian enclave with David. He also formed the core of the weekly trivia group, messaging everyone and showing up 30 minutes to an hour early to grab a table. I also watched shows and just went out for dinner with Nick, talking about fictional lore or other (thankfully) non-physics topics. It was through Nick I managed to worm my way into his undergrad friend group. Though I hope one day to join voice chat without Ari and John watching a twitch streamer, thanks for the fun times outside of that.

This research was funded by the NSF with the grants OAC 1550551 and OAC 2004157.

Table of Contents

Abstract

Current numerical codes can successfully evolve similar-mass binary black holes systems, and these numerical waveforms contributed to the success of the LIGO Collaboration's detection of gravitational waves. LIGO requires high resolution numerical waveforms for detection and parameter estimation of the source. Great effort was expended over several decades to produce the numerical methods used today. However, future detectors will require further improvements to numerical techniques to take full advantage of their detection capabilities. For example, the Laser Interferometer Space Antenna (LISA) will require higher resolution simulations of similar-mass-ratio systems than LIGO. LISA will also be able to detect extreme-mass-ratio inspiral (EMRI) systems. The EMRIs require a perturbative approach, and these techniques lags far behind numerical relativity. Improvements to current similar-mass codes and development of EMRI codes are necessary for future gravitational wave studies.

My first project improved the underlying framework of the Einstein Toolkit (ETK). I improved the ETK by implementing a new method for scheduling ghost zone synchronization and application of boundary conditions. The new approach reduces inter-processor communication overhead and improves the user experience. These improvements to the ETK improve its computational efficiency and enable users to more easily contribute to the collaboration.

I also implemented the first-order perturbative evolution equations for the EMRI system. This work builds on code for simulating the toy model of a particle with a scalar charge. This code differs from other time domain codes by evolving self-consistently by using the full self-force to provide a highly accurate waveform. I extended this code to be capable of evolving gravitational fields. I implemented even and odd master functions for the Regge-Wheeler-Zerilli gauge and verified convergence of the energy and angular momentum fluxes to frequency domain results. I derived and implemented evolution equations in the Lorenz gauge. The evolution equations in the Lorenz gauge are considerably more complicated than

the Regge-Wheeler-Zerilli gauge, and my code currently does not match the expected results. Still, my code is stable at long times and has effective constraint damping. These two codes represent significant progress towards the self-consistent evolution of the EMRI system at first order.

Chapter 1. Introduction

The past thirty years have seen significant advancements in gravitational physics, both in computational and experimental techniques. The development of advanced engineering solutions and new data analysis techniques culminated in the first gravitational wave detection by the LIGO Collaboration [1]. These detections rely on having high resolution waveforms from numerical calculations, especially for sky localization and parameter estimation. The future of gravitational wave astronomy will expand the range of detectable astrophysical systems and will require numerical waveforms for these systems. The next generation of ground-based detectors will also require higher resolution waveforms to fully extract all of the scientific data within the data stream, meaning current code must be improved to allow for faster generation of high resolution waveform templates. Particularly, the rise of multi-messenger astronomy means that gravitational wave signals must be identified with high accuracy so that the sky localization allows for correlation between gravitational wave data and data from other observatories (neutrinos with IceCube, light with traditional observatories, etc.). Thus, future detectors require numerical codes to perform higher resolution simulations with greater computational efficiency to fully extract all available scientific knowledge from these experiments.

In addition, proposed space-based detectors, such as the planned Laser Interferometer Space Antenna (LISA), will probe a different frequency range of gravitational waves than ground-based detectors like LIGO. This difference will allow for the detection of new types of binary systems, requiring the development of new numerical techniques to generate waveforms for these systems. One system of interest for LISA is the extreme-mass-ratio inspiral (EMRI). The EMRI system consists of a compact object (with mass of order $10M_{\odot}$) moving around a supermassive black hole (with mass of order $10^6 M_{\odot}$). These systems are particularly interesting because EMRIs have very long inspiral times, and LISA is expected to be capable of detecting an individual inspiral for millions of orbits over several years. Measurement of these systems will provide unique tests for general relativity in the strong-gravity regime.

However, these long inspirals mean that several EMRI signals will almost certainly overlap, and highly accurate waveform template catalogs are required for matched template filtering and binary parameter estimation. While binary black hole mergers have been modeled at high accuracy for similar-mass systems, the inspiral and merger in the extreme-mass-ratio limit has yet to be completely resolved. Several scaling features of binary black hole systems make EMRIs entirely unapproachable from techniques used for similar mass systems. The dynamic timescale for the smaller object is proportional to the mass ratio, while the time to merge is proportional to the inverse mass ratio. In addition, the spatial scale required to resolve the smaller object sets a time step via the Courant-Friedrichs-Lewy (CFL) condition which is far smaller than would otherwise be needed. All of these conspire to make traditional numerical relativity techniques unusable in the extreme-mass-ratio limit. These issues required the development of a new approach specifically for EMRIs, called the self-force approach. This method uses perturbative techniques to expand the space-time metric by order in the mass ratio.

I have worked on two projects for advancing waveform generation in preparation for future gravitational wave detectors. I improved the Cactus Framework, which is the underlying foundation of the Einstein Toolkit, an open-source codebase for doing numerical relativity simulations. In this work, I improved the scheduling to reduce communication—reducing runtime as a result—and simplify the user experience. I have also developed code for performing EMRI simulations using the self-force approach. I implemented gravitational self-force code in two different gauges, the Regge-Wheeler-Zerilli gauge and the Lorenz gauge. In Chapter 2, I introduce the Cactus Framework and describe my improvements to its task scheduling. In Chapter 3, I examine the extreme-mass-ratio inspiral in detail and the many techniques that have been developed to simulate this system. I present my contributions and results for my code in Chapters 4 and 5 for the two different gauges.

Chapter 2. Synchronization Automation in the Cactus Framework

2.1. Einstein Toolkit

The Einstein Toolkit (ETK) is an open-source suite of computational tools for numerical astrophysics simulations [3]. This toolkit is used for simulating a wide range of astrophysical systems. While there are some groups which use the ETK for other research, most using the toolkit are simulating neutron stars, black holes, and compact object binary mergers. Excluding the self-force code by Peter Diener and myself, all code within the ETK is built on the Cactus Framework (Cactus). Cactus is an open-source environment for numerically solving Cauchy problems on a compute cluster [4]. While Cactus is used as the core framework for several codes, it was designed primarily for simulations of gravitational physics systems. Cactus implements many basic requirements for numerical simulations, such as creation of distributed data structures, I/O, checkpointing, and parallelism. As it is a 'cactus', the core code which handles these features is referred to as the flesh, and the application modules which perform numerical calculations or do other tasks are called thorns. These thorns can be designed and tested on workstations and then easily run on clusters. These thorns provide the necessary information for Cactus to connect them to other thorns. The core paradigm of Cactus is to allow for contributors to focus on writing physics code while many of the complications of running on clusters is offloaded to Cactus. This reduces the need to reinvent the wheel and also allows for optimizations made to Cactus to benefit the research of many different research groups.

Within a thorn, scheduling and boundary condition application information is also provided by the programmer for each scheduled subroutine. Cactus provides a basic workflow engine in which thorns use a domain specific language (DSL) to schedule subroutines to run at startup, at each time step, at after or before other subroutines, etc. Using this mechanism,

various thorns are able to coordinate their activities in a modular fashion. For example, I/O routines can be scheduled to run after each time step and save data, and initial data routines can be scheduled to run at startup. Cactus supports both Fortran and C/C++ code. Using the DSL, Cactus is told the language of each subroutine and automatically implements the proper linking so that the subroutines can communicate.

One performance issue in Cactus is the scheduling of ghost zone synchronization. Ghost zones are a numerical technique used to resolve a problem in parallelized simulations. Most algorithms for evolving physical systems (e.g. finite differencing) require data from adjacent points on the numerical grid. When using muliple processes, the grid is split between the MPI ranks, the local grid for a single processor does not have the data for grid points along the processor boundaries. The region of the grid which is assigned to another processor but required in order to evolve the local grid points is referred to as a ghost zone. To make this adjacent data available, a copy of the ghost zone data is stored on the local memory. A simple diagram of this is shown in Figure 2.1.

However, this local data must be updated to new values as the other processors evolve their grid points. Thus, the ghost zone information must be synchronized at certain points in the code. The Cactus scheduling system relies on the manual synchronization of ghost zones for grid functions (i.e. distributed matrices). Using the DSL, the scheduled subroutine must declare the set of grid functions to synchronize after it runs. Unfortunately, deciding which subroutines should synchronize which grid functions is a non-trivial problem requiring an in-depth understanding of how the core framework works. This problem is worsened because of the ETK's modular nature. Compiling the ETK requires providing a list of all thorns that are present and need to be compiled and linked. A single simulation never needs every thorn, so thorns are turned off or on at runtime based on the parameter file used to initialize the simulation. Since the scheduling information of a subroutine is determined by a compile-time declaration, thorn writers must determine a single correct choice of synchronizations for all the possible thorn combinations. Incorrect synchronization can result in

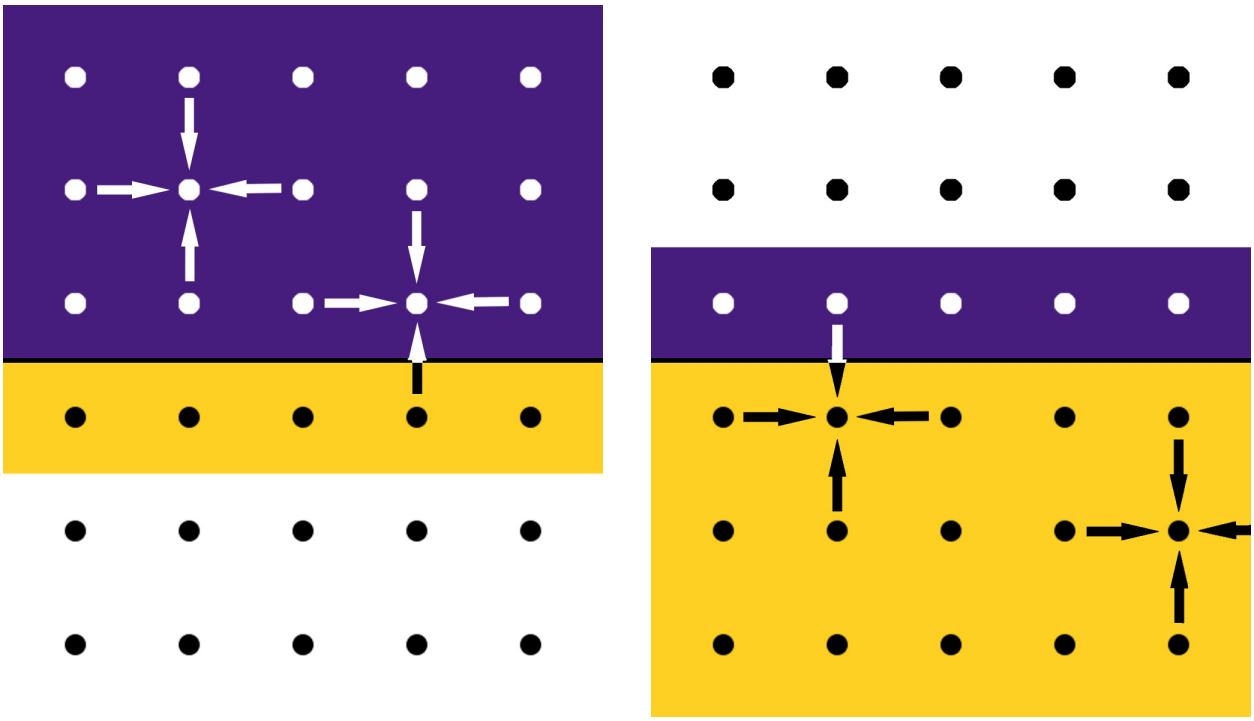

Figure 2.1. Example of a grid split across two processors. On the left, the first processor has its local grid (in purple) and a local copy some gridpoints from the second processor (in yellow). The section in yellow is the ghost zone. On the right, the second processor has its local data (in yellow) and its ghost zone (in purple).

over-synchronization (a performance problem) or under-synchronization (a numerical error). Currently, programmers using Cactus have no tools to help them determine whether their thorns are over-synchronizing or failing to synchronize. The case of under-synchronization can be detected because the resulting simulations will accrue errors, but tracking down the source can prove difficult. The difficulty of properly scheduling sync statements also creates a barrier for users and thorn writers, especially new thorn writers who must become familiar with the internal workings of Cactus before they can contribute to the research community.

In addition to synchronization issues, boundary conditions are handled separately from synchronization in Cactus. This is unfortunate, because both synchronization and boundary conditions are mechanisms for filling in the outer cells of distributed grids and are naturally applied at the same time. The `Boundary` thorn in Cactus handles physical boundary conditions, while the `SymBase` thorn manages symmetry boundary conditions. Regardless

of the type of boundary, however, boundary updates are scheduled manually. Many thorns also introduce additional boundary condition subroutines via internal methods. Scheduling of the subroutines provided by the `Boundary` thorn—and, by extension, any subroutines registered with `SymBase`—is also left to the individual thorn writer. This paradigm leads to similar issues as with synchronization. The implementation of ghost zone synchronization and application of boundary conditions in Cactus both place an unnecessary burden on thorn writers and increase the risk of numerical errors in simulations.

These operations are further complicated by the use of adaptive mesh refinement (AMR). With a unigrid simulation—that is, one not using AMR—the grid spacings are determined by the region with the need for the highest resolution. In astrophysical systems, this normally means around the star or black hole. However, this degree of resolution is entirely unnecessary away from the object, so this wastes computational resources and in many cases would simply make the simulation impossible to perform. AMR begins with a coarse grid across the domain, and then defines refinement levels in the regions where a finer grid is required. This process can be done multiple times, with each new level covering a smaller part of the domain and having a smaller grid spacing. The 'adaptive' part of AMR allows for the introduction of a new mesh refinement level during the simulation. If the numerical errors grow too large, a new level can be introduced to remedy this.

Most simulations with the ETK require a special type of thorn called a "driver thorn" to manage the grid and provide AMR. `Carpet` is the most commonly used driver thorn and provides a Berger-Oliger style [5] of adaptive mesh refinement. AMR introduces two new data movement operations in addition to synchronization of ghost zones: prolongation of data from coarser grids to finer grids and restriction of data from finer grids into coarser grids. `Carpet`'s AMR implementation uses smaller timesteps on fine grids than on coarse grids (it uses an optimal Courant-Friedrichs-Lewi condition between spatial resolution and time step). Prolongation involves interpolation in time, necessitating retaining data from previous timesteps. These operations interact in complex ways with boundary conditions,

typically requiring that physical boundary and symmetry boundary conditions be applied after each operation. Under certain circumstances, more than one application of the boundary conditions may be required, e.g. such as when a fine grid catches up with a coarse grid. This event triggers (in order): prolongation from coarse to fine to fill in the outer regions of the fine grid; restriction from fine to coarse grid to overwrite data on the inner region of the coarse grid with data from the fine grid; and, finally, prolongation from coarse to fine grid again because the source region of the prolongation operator overlaps with the restricted region. All of these subtleties are well understood by `Carpet` which, however, lacks direct control over when boundary conditions are applied and has to rely on thorn writers to schedule appropriate subroutines instead.

2.2. The PreSync Project

I developed a new scheduling and synchronization method for Cactus to improve the programmability and efficiency of the Einstein Toolkit. The new methodology takes a more data-driven approach to the computation of data in the boundary and ghost zones of the grid (computing that data only when it is needed), a first step toward making the framework as a whole more data-driven, following a trend of many modern frameworks. For example, the Cello Astrophysics code [6] is a re-write of Enzo [7] based on Charm++ [8], an asynchronous multi-task framework. As another example, the Octo-tiger code [9], which simulates interacting binary stars, was a re-write of an older MPI code that leverages the HPX Framework [10] to use futures. Both Charm++ and HPX track data validity and only allow code to execute when the requested data is ready.

As part of this project (referred to hereafter as *PreSync*), I also integrate the application of boundary conditions into synchronization. Synchronization and application of boundary conditions are now handled automatically by Cactus and are applied at the same time. The new approach requires each subroutine to have "read" and "write" declarations for individual grid functions. The read/write declarations for grid functions identify the region of validity. PreSync tracks the interior, boundaries, and ghost zones independently, so all combinations

of validity are possible. However, for normal thorns the only declarations needed are usually "interior" or "everywhere." The "write" declarations change the grid function's region of validity, and "read" declarations require the grid function to be valid for the specified region. Before scheduled subroutines run, Cactus checks these declarations and, if a grid function is only valid on the interior but is needed everywhere, performs synchronization and applies boundary conditions as needed.

This method removes the difficulty of deciding where synchronization should take place and removes unneeded synchronizations. Programmers need only declare what grid functions a given subroutine uses and on which parts of the grid they are read or written. The PreSync project also modifies Cactus to include all the infrastructure needed to handle boundary conditions; thorns which provide boundary conditions simply register their boundary conditions with Cactus. These changes also decouple the scheduling information of different thorns; where scheduling subroutines previously required knowing how that subroutine would interact with others in the schedule tree (determined at runtime), scheduling subroutines with PreSync only requires knowledge of that one subroutine and what it reads/writes. By internalizing the details of synchronization and boundary condition application, I remove unnecessary complications from thorn writers using Cactus effectively.

2.3. Compatibility and Advanced Features

In a large collaborative project such as the ETK, backward compatibility is important. As such, PreSync does not prevent the old synchronization mechanism from functioning, i.e. manual synchronization can still be used. PreSync has also been incorporated into the legacy code, allowing for easier comparisons between the two methods. Once a thorn has been properly updated to use PreSync, a single runtime flag allows for switching between the old and new methods. This allows for quick and easy code comparison to verify that PreSync has been properly implemented into an updated thorn.

2.3.1. Runtime-Based Read/Write Declarations

While the subroutines of most thorns have simple read/write declarations, some subroutines have more elaborate behavior which needs more advanced treatment. As an example, `GRHydro`, the hydrodynamics evolution thorn in the Einstein Toolkit, has parameters which can be changed at runtime that decide whether magnetic fields should be evolved. The read/write declarations for a subroutine are normally given at compile time, which means that the declarations themselves cannot take this choice into account. To resolve this discrepancy, logical "if statements" are used in the scheduling, with the same subroutine being scheduled with different read/write declarations depending on the state of the magnetic field parameter. This was already being done in `GRHydro` to schedule different subroutines depending on whether magnetohydrodynamics was active or not. However, some of `GRHydro`'s subroutines instead use an if statement within the code that depends on a runtime parameter. This case requires some extra work to change the scheduling information, but most thorns using the ETK can be transitioned relatively easily.

The I/O thorns exhibit even more dynamic behavior—any number of grid functions could be chosen at runtime to write to files. Most grid functions are valid "everywhere" when the simulation reaches the output stage. However, some thorns have output grid functions which are only written and never read. Since they are never read, synchronization and boundary condition application never triggers. To handle this case, or any other situation in which a grid function's ghost zones are not synchronized, the I/O thorns need to call special subroutines to check the region of validity and trigger synchronization if necessary. PreSync provides subroutines for this purpose, and the I/O thorns in Cactus have been updated to use these features.

The AMR within `Carpet` also demands special treatment to properly interface with PreSync. The prolongation operation in AMR requires that data on its coarse grid source be valid "everywhere," yet this data is only required once requested by the fine grid. PreSync

thus recursively checks if coarser levels need to be synchronized or have prolongation applied to them just before synchronizing a finer level.

2.3.2. Diagnostics

The greatest burden for the transition to PreSync is adapting old thorns to use the new system. To assist with this endeavor, I provide various systems to provide feedback and error checking for read/write declarations. These features also benefit current thorns writers during and after the transition into the new method. Currently, Cactus gives all subroutines read and write access to all grid functions. I employ automatic code generation to create macros which instead limit the grid functions accessible to each subroutine. These restricted macros are generated for each scheduled function and only provide access to grid functions declared via read/write specifications. Therefore, the use of any grid functions in the subroutine's source not given in the read/write declarations will result in a compilation error. Additionally, read-only grid functions are declared as constants, so writing the grid function will also cause a compilation error. Cactus also sets the pointers to undeclared grid functions to null when they should not be accessed, triggering runtime segmentation faults when attempts to access grid functions which have not properly been declared.

The new macros can fail to detect a problem when a subroutine calls another subroutine outside the normal Cactus scheduler, as the compiler does not know if the called subroutine reads or writes the grid function within it. If C/C++ code explicitly passes the grid functions, then the macros will behave as normal. However, some application thorns use the macros within internal subroutines, which prevents this safety mechanism from working.

Currently, scheduled Fortran code cannot currently use the new macros, as Fortran requires an explicit argument list, and Cactus passes the full list of variables. New Fortran macros and argument lists are generated correctly, but the subroutine caller in Cactus must be updated to use these as well. However, they can be used at compile time for debugging and then reverted to the original once the code successfully compiles.

To provide a check for correct region access (i.e. "interior" vs. "everywhere"), I have created thorn named `ReadWriteDiagnostics` to determine which grid functions are written by each subroutine at runtime. This thorn makes use of a `Carpet` feature that allows special subroutines to be run before and after each regularly scheduled subroutine, providing something similar to "aspect-oriented programming."

`ReadWriteDiagnostics` uses checksums, computed both before and after a routine runs, to determine whether a grid function has been written, and whether it was written in the "interior" or "everywhere." There are some subroutines which write grid functions lazily by writing everywhere, knowing that the data written in the boundaries will be incorrect and relying on another subroutine or an explicit synchronization will overwrite the "bad" data. This behavior will, of course, create a false "everywhere" in the diagnostic tool. The code may "write everywhere," but the data is only valid on the interior. Ideally, thorn writers should not do this, but in the past it was not a problem. In these cases, the diagnostic tool will not be able to assist in error-checking.

2.4. Results

PreSync has been implemented in Cactus and the Einstein Toolkit, and the vast majority of the test suite passes. Remaining tests rely on thorns which have not yet been converted to be PreSync-ready. In addition, all the tests pass if PreSync is turned off, showing that backward compatibility has been properly implemented. In addition, comparative tests using the Einstein Toolkit have been run with the old and new methods, and PreSync synchronizes approximately five percent less often. I ran the tests on two different machines: Melete and SuperMike. On Melete, these changes lead to a speedup of four percent on two processors and nearly twenty percent on thirty-two processors. The speedup on SuperMike is less consistent but falls within the range of two to twenty percent improvement. For all of the collected data, the plots show runtimes for both the old and new methods using various numbers of processors. The relative difference plots show the relative difference of the runtimes ($\frac{old-new}{old}$) for each of the processor values.

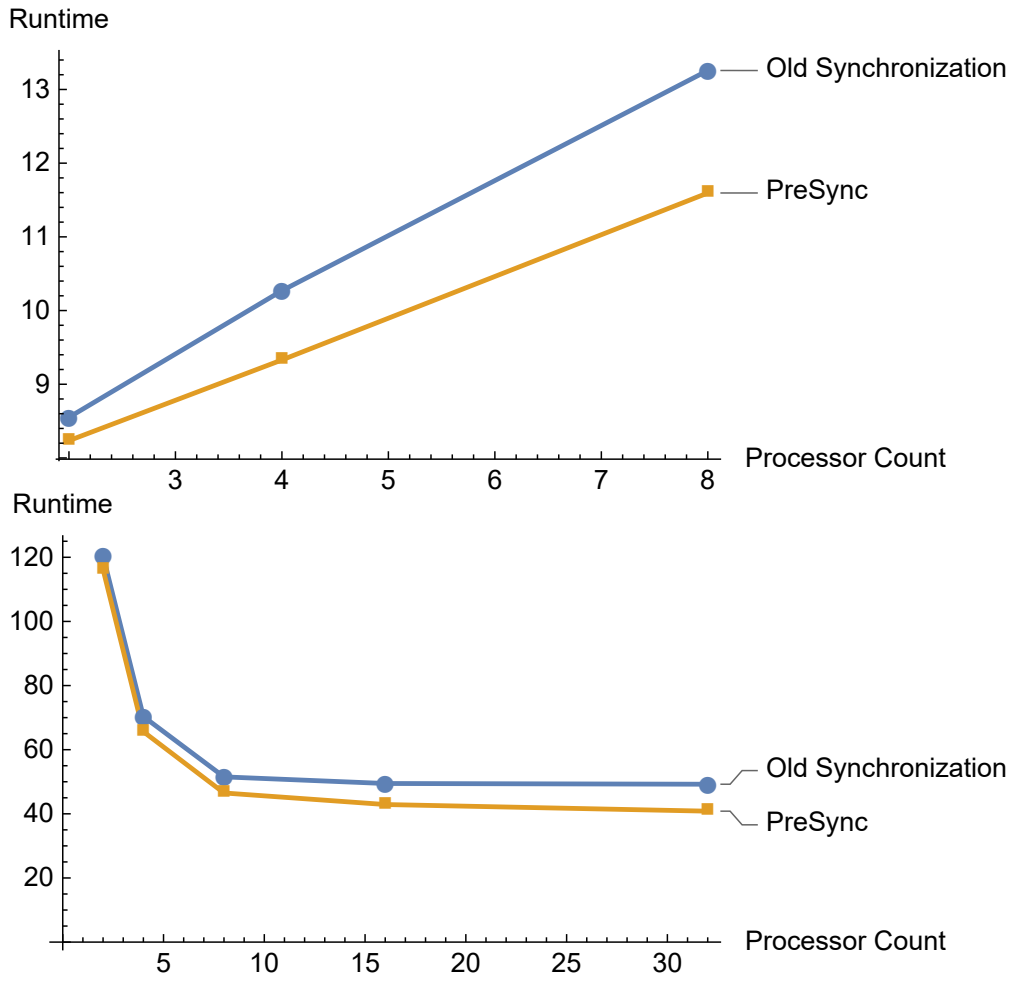

Figure 2.2. Runtime comparison of Cactus and PreSync Cactus Unigrid simulations with 10^3 (top) and 50^3 (bottom) grid points on Melete. The small grid test is too small to benefit from parallelism, while the larger grid test is of sufficient size to benefit from parallelism.

The results for strong scaling tests using 10^3 and 50^3 numerical grids are shown in Figures 2.2 and 2.3. These tests were performed on the LSU Melete cluster. In both tests, there is a clear runtime improvement with PreSync that increases with processor count. PreSync takes four percent less time with two processors and seventeen percent less time with thirty-two processors, showing significant improvements over the old synchronization method. The 10^3 grid test only goes to eight processors because there was not enough useful work to support more. While this specific test does not benefit from parallelism, it clearly benefits from PreSync. As the processor count increases, PreSync's runtime improvement becomes more significant, as expected.

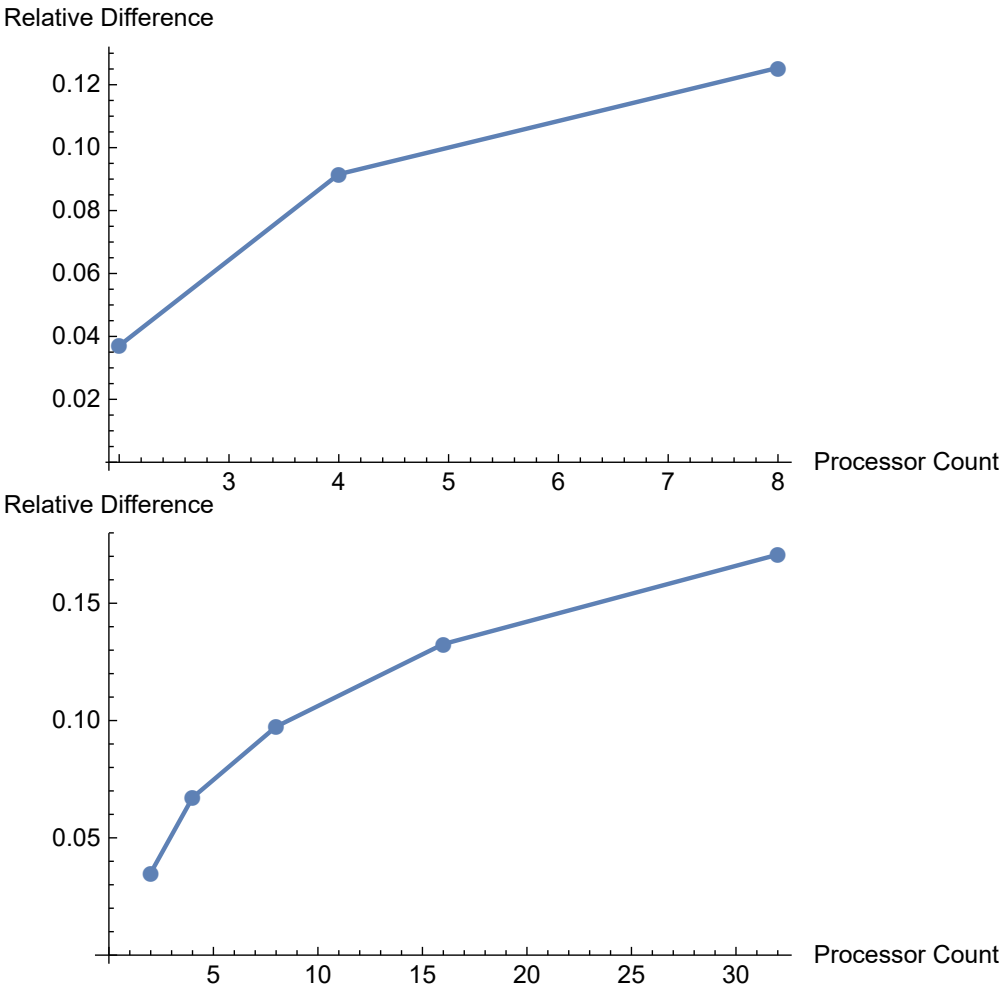

Figure 2.3. Relative difference between Cactus and PreSync Cactus Unigrid simulations with 10^3 (top) and 50^3 (bottom) grid points on Melete. The runtime is 3-20% shorter for PreSync, depending on the number of processors.

The 50^3 grid test goes up to thirty-two processors on Melete. This test is large enough that it benefits from the additional processors, and the parallelism-based improvements are clearly visible. In addition, PreSync shows visible improvements over the old synchronization method, as the percentage improvement on Melete continues to increase with increasing processor count.

I also ran two strong scaling tests on SuperMike: the same 50^3 grid point test as on Melete and an even larger 100^3 grid point test. I used up to sixty-four processors (4 nodes) on SuperMike for both tests. Figures 2.4 and 2.5 show the results from these tests. PreSync outperforms the old method in every test and provides significant improvements over the old

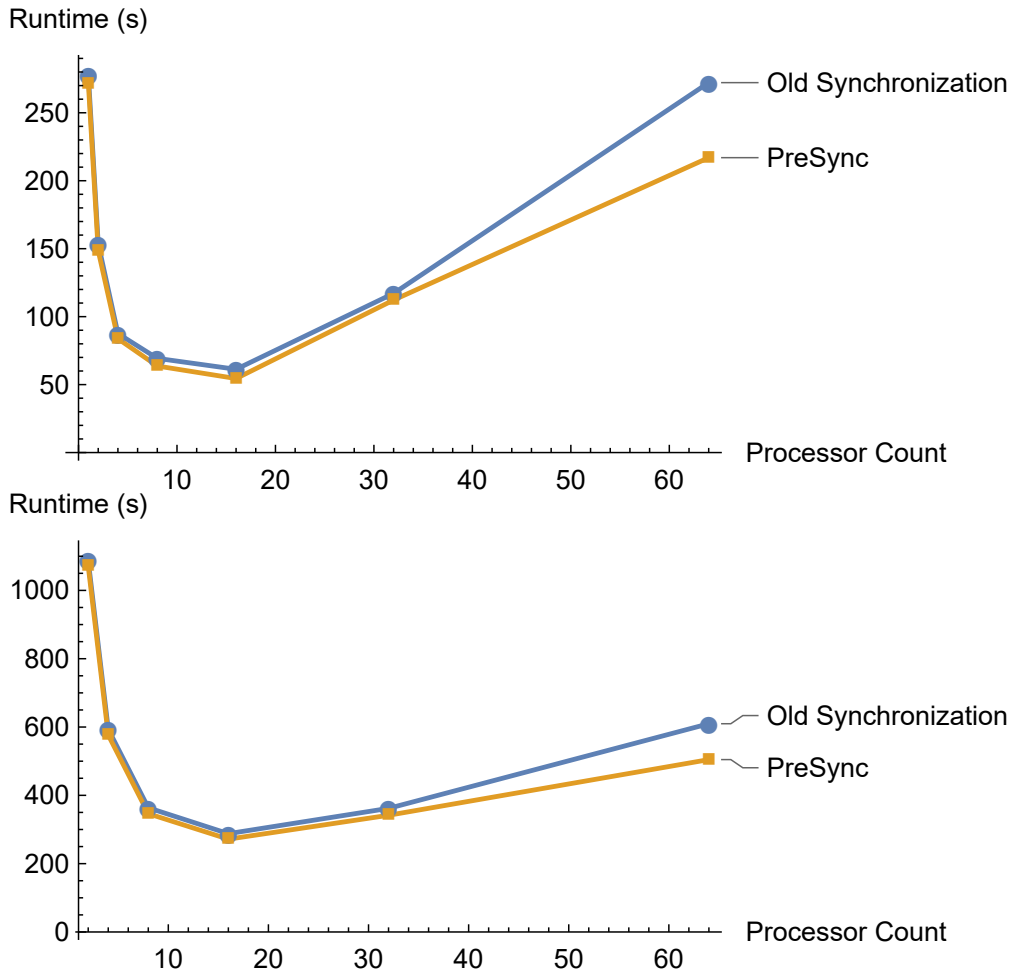

Figure 2.4. Relative difference between Cactus and PreSync Cactus Unigrid simulations with 50^3 (top) and 100^3 (bottom) grid points on SuperMike. The runtime is 3-20% shorter for PreSync, depending on the number of processors.

method with high processor count. The behavior change after sixteen processors is because each node of SuperMike contains sixteen processors, so the larger runs introduce inter-node communication in addition to inter-processor communication. The runtime speedup is between two and twenty percent and increases with processor count. The transition to multiple nodes stalls or reduces the improvement, but the pattern continues after this transition.

These tests were performed on Unigrid, so the performance with adaptive mesh refinement (AMR) is undetermined. However, these results already show that PreSync is more computationally efficient than the old synchronization method. While the effect on AMR

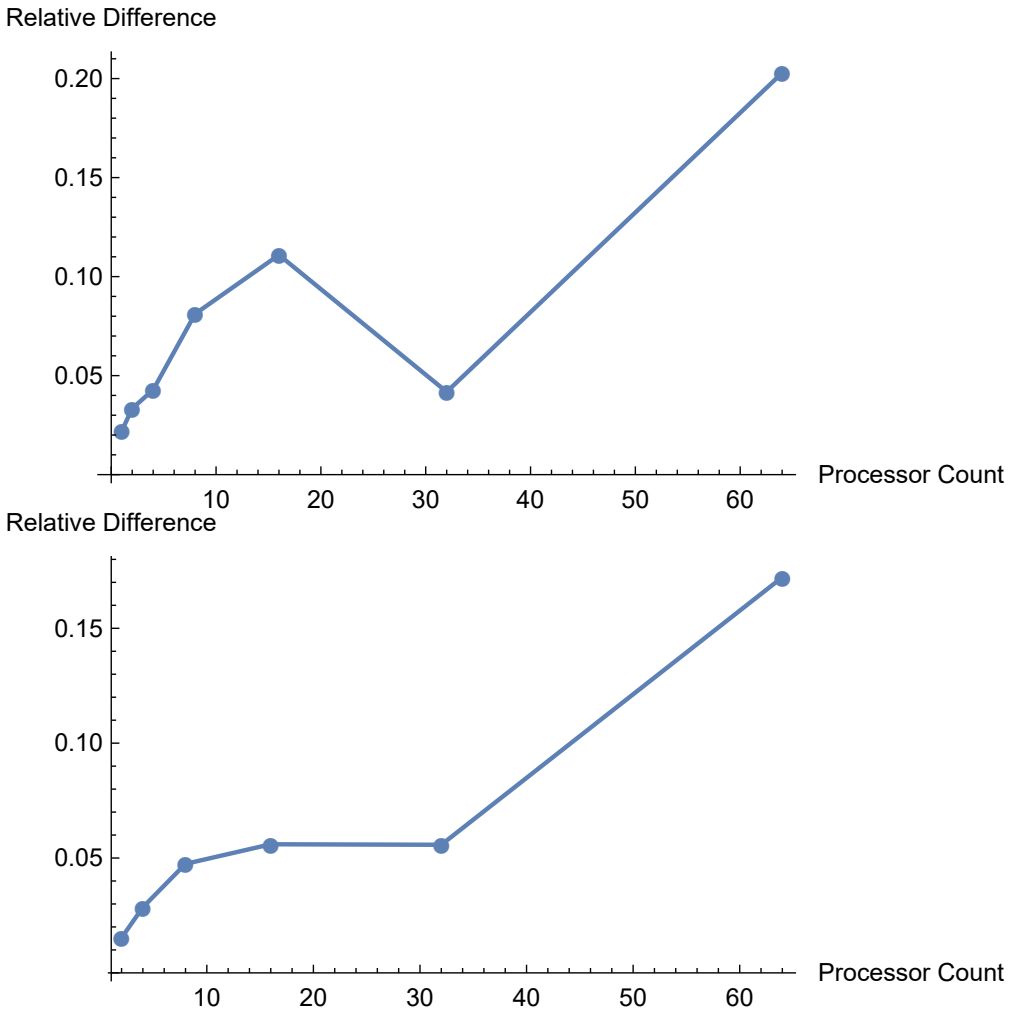

Figure 2.5. Runtime comparison of Cactus and PreSync Cactus Unigrid simulations with 50^3 (top) and 100^3 (bottom) grid points on SuperMike. The increase at 32 processors is because a single node on SuperMike only has 16 processors, so inter-node communication slows down the test.

simulations is not yet quantified, the preliminary AMR testing shows reduced synchronization as well, meaning that production-level simulations will likely see speed improvements from PreSync.

The PreSync project makes programming Cactus easier and provides noticeable speed improvements. Scheduling a subroutine only requires knowledge of that one subroutine, removing the need for thorn writers to understand the details of how a routine fits into the schedule as a whole. Also, deciding the read/write declarations for a subroutine is usually a straightforward process, while the proper placing of sync statements is frequently a non-

trivial problem. For backward compatibility, PreSync also allows for the old synchronization method to be used, and parameters exist to switch between these methods at runtime.

Chapter 3. Extreme-mass-ratio Inspiral: The Self-Force Approach

3.1. EMRI Basics

As previously mentioned, the EMRI system suffers from several unfortunate scaling issues which prevent the use of traditional numerical relativity techniques. The smallest dynamic timescale of physics in the evolution is proportional to the mass ratio, which means that the numerical time step required to resolve the physics decreases as the mass ratio decreases. Simultaneously, the total time till merger is proportional to the inverse mass ratio. Finally, the spatial scale required to resolve the physics around the smaller object also sets a maximum time step via the Courant-Friedrichs-Lewy (CFL) condition. All of these conspire to make traditional numerical relativity techniques unusable in the extreme-mass-ratio limit.

Fortunately, the small mass ratio which causes these issues also provides an alternative approach. Instead of evolving the system with the full metric $\tilde{g}$, we may consider a background metric g generated by the larger object perturbed by the smaller object—which we hereafter refer to as the particle. The perturbation is expanded in the mass ratio $\epsilon = \frac{m}{M}$. Then, the full metric is given by

$$\tilde{g}_{\alpha\beta} = g_{\alpha\beta} + \epsilon h^1_{\alpha\beta} + \epsilon^2 h^2_{\alpha\beta} + \mathcal{O}(\epsilon^3) \tag{3.1}$$

where $h^n_{\alpha\beta}$ is the nth order perturbation metric. To zeroth order, the particle follows a geodesic in the background spacetime. At first order, the particle instead follows a geodesic in the perturbed spacetime $g_{\alpha\beta} + \epsilon h^1_{\alpha\beta}$. Unless explicitly mentioned, we henceforth only consider the first order perturbation and drop the superscript on h. By mapping the geodesic in the perturbed spacetime to a path in the background spacetime, we can find equations of motion for the particle in the background spacetime:

$$a^\mu = -\frac{1}{2}(g^{\mu\nu} + u^\mu u^\nu)(2h^R_{\nu\lambda;\rho} - 2h^R_{\lambda\rho;\nu}). \tag{3.2}$$

Note that the perturbation field in Equation 3.2 is the 'regular' field h^R; this subtle change from the 'retarded' field h to the 'regular' field h^R is necessary in order to have a finite quantities at the location of the particle. The details of the separation of the full perturbation into a regular and singular piece are discussed in the following section. The equations of motion for a point mass were first derived by Mino, Sasaki, and Tanaka [11] and Quinn and Wald [12] and is sometimes referred to as the MiSaTaQuWa equation. An immediate concern for these derivations is whether a general compact object behaves identically to the point mass used in the derivation of the MiSaTaQuWa equation since internal dynamics could play a part in the evolution. However, later work showed that these equations were valid for any sufficiently compact object, regardless of internal structure [13–15]. These publications laid the groundwork for the self-force approach, which is the method of evolving the (accelerating) particle in the background spacetime by treating the perturbative field as an external force acting on the particle. This acceleration is due to the particle's own gravitational field, hence the name 'self-force'.

3.2. Radiation Reaction

While the self-force approach provides a promising avenue for evolving EMRIs, problems quickly arise in computing these quantities. In order to calculate the force acting on the particle, the perturbed metric h must be evaluated at the location of the particle. Of course, the field generated by any point source is infinite at its own location, so this immediately causes significant trouble. Therefore, a method must be developed to extract a non-singular expression for the particle's own field at its location. The problem has many similarities to radiation reaction in electromagnetism, so we may examine this problem for inspiration.

As a simple example, consider a charged point particle moving around a (fixed) central point. The particle produces an electromagnetic vector potential satisfying the wave equation

$$\Box A^\alpha = -4\pi j^\alpha \tag{3.3}$$

where j^α is the current density. The particle will emit outgoing radiation due to its motion. Therefore, radiation reaction should drive the particle to spiral towards the center. However, the current density of the point particle is a four-dimensional Dirac delta function, meaning that the right-hand side of Equation 3.3 is infinite along the particle's worldline. The force on the particle due to radiation reaction is from its own field, and the singular nature of the wave equation at the particle's position prevents the immediate calculation of this force.

To determine the effect of the particle's field on its own motion, a non-singular expression must be found which fully encapsulates the effect of radiation reaction on the particle's trajectory. we shall refer to the specific solution to Equation 3.3 with outgoing radiation and an inward-spiralling particle as A^α_{ret}, called the retarded solution. There, of course, also exists a time-reversed "advanced" solution with ingoing radiation and an outward-spiralling particle, denoted as A^α_{adv}. From these, we can construct a time-invariant solution

$$A^\alpha_S = \frac{1}{2}(A^\alpha_{ret} + A^\alpha_{adv}) \tag{3.4}$$

This solution has equal amounts of ingoing and outgoing radiation, canceling the effect on the particle and causing it to simply orbit instead of spiraling. While this solution exerts no force on the particle, all three solutions are equally singular along the particle's worldline. Therefore, the singular nature of A^α_{ret} may be removed by subtracting away A^α_S:

$$A^\alpha_R = A^\alpha_{ret} - A^\alpha_S = \frac{1}{2}(A^\alpha_{ret} - A^\alpha_{adv}) \tag{3.5}$$

Since A^α_{ret} contains the full effects of the radiation reaction and subtracting A^α_S removes the singularity without introducing any new forces, A^α_R must be the (fully regular) field which is responsible for the force on the particle. The singular field A^α_S has been rigorously shown to not exert any forces on the particle [16, 17].

Unfortunately, spacetime curvature further complicates the application these results. By using Green's functions, one can derive the dependencies of these solutions on the particle's

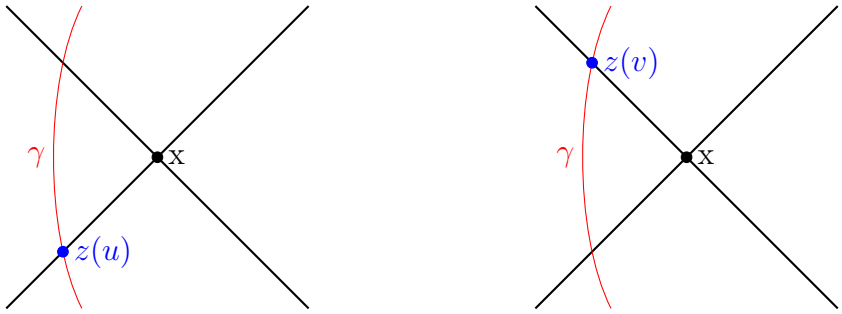

Figure 3.1. The retarded field at spacetime point x depends on the particle's motion at spacetime point $z(u)$ along the particle's worldline γ. Similarly, the advanced field at x depends on the particle's motion at $z(v)$.

worldline γ [17]. Here, I present a summary of these derivations. In Minkowski spacetime, the retarded (advanced) field only has support on the past (future) light cone. For a spacetime point x, the retarded field is generated by the particle at the retarded point $z(u)$, where u is the time parameter for that point on γ. Similarly, the advanced field is generated by the particle at the advanced point $z(v)$ with time parameter v. Figure 3.1 shows these relationships. To calculate the radiation reaction, the point x is taken to be on the worldline. In the limit as x approaches the worldline γ, $z(u) \to z(v)$ and thus the force due to radiation reaction depends only on the particle's current state.

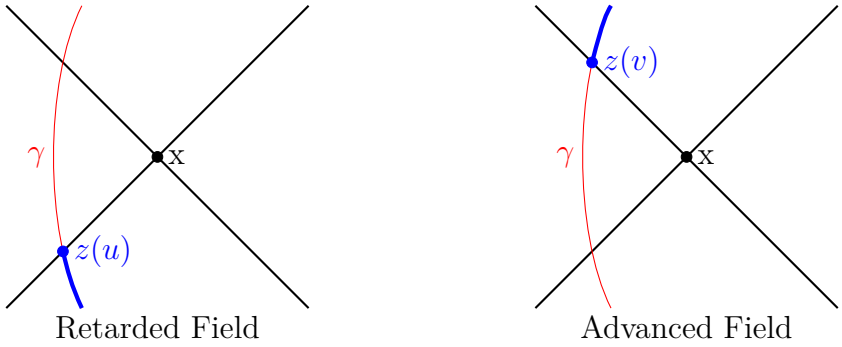

Figure 3.2. The retarded field at spacetime point x depend on the particle's motion for its entire past history. Similarly, the advanced field at x depends on the particle's motion for its entire future history.

In contrast, the retarded field in curved spacetime depends on the particle's entire past history from time u backward to past time-like infinity. This occurs because of several

reasons. In flat spacetime, radiation reaction only affects the particle the instant that the radiation is emitted. In curved spacetime, emitted radiation can curve around the central body and intersect with the particle's worldline at a later time. Also, curved spacetime causes backscattering of the radiation off of the curvature, which travels back towards the particle. These additional sources all contribute to the retarded field. Similarly, the advanced field depends on the particle's entire *future* history from $z(v)$ to future time-like infinity. Figure 3.2 shows these relationships. If x is taken to be on the particle's worldline, then the regular field A^{α}_{R} will depend on the particle's entire history, past *and* future! Using the symmetric field as defined in Equation 3.4 would therefore introduce noncausality to the equations.

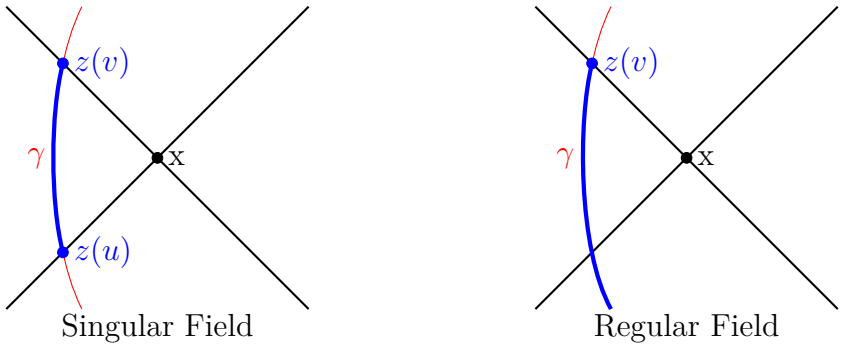

Figure 3.3. The singular field at spacetime point x depends on the particle's motion in the temporal interval $u \leq \tau \leq v$. The regular field at x depends on the particle's motion in the temporal interval $-\infty \leq \tau \leq v$.

A solution known as the Detweiler-Whiting Decomposition [18] resolved this problem by introducing a new two-point function into the singular Green's function which allows for a clean separation of the singular and regular pieces of the field without any dependencies on the particle's future. The singular field at a spacetime point x depends on the particle's motion in the temporal interval $u \leq \tau \leq v$. The regular field at x depends on the particle's motion in the temporal interval $-\infty \leq \tau \leq v$. These relationships are shown in Figure 3.3. In the limit as x approaches γ, the regular field depends on the particle's entire past history, and the singular field only depends on the particle's instantaneous motion. The regular field—and by extension, the self-force—still depends on the particle's entire past history,

which makes initial conditions difficult to determine. I tackle the issue of initial conditions later, in Section 3.7.

3.3. Mode-sum Decomposition

While the Detweiler-Whiting decomposition was a significant development in the field, it alone is not sufficient to allow for numerical computation of the self-force. Mode-sum decomposition was developed so that these singular quantities could be expressed as sums of finite quantities for numerical computation. The combination of the Detweiler-Whiting and mode-sum decompositions provide a reasonable method to calculate the regular field.

To explain the mode-sum decomposition, I use the case of a particle with scalar charge q. The particle's potential Φ satisfies the equation

$$\Box\Phi = -4\pi q \int_{\gamma} \frac{\delta^{(4)}\left(x - z(\tau)\right)}{\sqrt{-g}} d\tau \tag{3.6}$$

where, as before, $z(\tau)$ represents a parameterization of the particle's worldline. From this, the scalar self-force is

$$F_{\alpha} = q\nabla_{\alpha}\Phi_{R} = q(\nabla_{\alpha}\Phi_{ret} - \nabla_{\alpha}\Phi_{S}) \tag{3.7}$$

However, calculating these quantities directly proves difficult for several reasons. First, the retarded field depends on the particle's entire past history, which is not known. Second, both potentials diverge at the particle's location, meaning a finite representation must be found to perform this subtraction. The mode-sum method does this by decomposing the field into spherical-harmonic modes

$$\Phi = \sum_{\ell m} \Phi^{\ell m}(t, r) Y^{\ell m}(\theta, \phi) \tag{3.8}$$

This decomposition expresses the divergent field in terms of an infinite sum of finite fields; in the case of spherically symmetric background spacetimes, this also produces completely decoupled evolution equations for each mode. Since the retarded field depends on the particle's

entire past history, the expression

$$(\nabla\Phi)_\ell = \sum_m \nabla_\alpha \left[\Phi^{\ell m}(t,r)Y^{\ell m}(\theta,\phi)\right] \tag{3.9}$$

must be computed directly. The singular field, however, is only dependent on the particle's current motion. It can therefore always be expanded in powers of distance to the particle. Thanks to this, multipole decomposition may be used to more quickly calculate the singular component of the field. Taking all this into account,

$$\begin{aligned} F_\alpha \approx q\sum_{\ell=0}^{L}\Bigg[&(\nabla\Phi)_\ell - (\ell+\frac{1}{2})A_\alpha - B_\alpha - \frac{C_\alpha}{\ell+\frac{1}{2}} - \frac{D_\alpha}{(\ell-\frac{1}{2})(\ell+\frac{3}{2})} \\ &- \frac{E_\alpha}{(\ell-\frac{3}{2})(\ell-\frac{1}{2})(\ell+\frac{3}{2})(\ell+\frac{5}{2})} - \cdots\Bigg] \end{aligned} \tag{3.10}$$

where L is the maximum ℓ in the truncated sum and A_α, B_α, and so on are the (ℓ-independent) functions from the multipole decomposition of the singular field. The polynomials these 'regularization parameters' represent have been determined [19] , and all terms after C_α evaluate to zero in the full sum from $\ell = 0$ to $\ell = \infty$. However, the D_α and E_α terms are included in the truncated sum when possible, as they improve the convergence rate of the ℓ-mode sum. Unfortunately, these parameters are not known in general. The regularization parameters have been calculated for orbits in Schwarzschild spacetime only up to D_α [19, 20]. In Kerr spacetime, the parameters are only known up to C_α [21]. However, additional parameters have been estimated by fitting numerical results to Equation 3.10.

3.4. Self-force Methods

Several different methods have been developed to evolve EMRIs using the self-force, each with its own advantages and drawbacks. Most modern techniques incorporate both the Detweiler-Whiting and mode-sum decomposition in some form. Some of the most well-developed codes transform the equations from the time domain into the frequency domain. Several frequency domain codes exist which can calculate the self-force for generic orbits

to very high accuracy [22, 23]. However, frequency domain codes have two handicaps. The lesser problem is that they are eccentricity-limited—more eccentric orbits require more and more frequency modes to accurately capture the motion, and the number of modes eventually becomes untenable. More critically, they are restricted to periodic orbits, which means that they cannot directly evolve the inspiral. Despite this, they provide a method to validate new codes. The validity of time domain codes can be partially assessed by forcing the time domain simulation to follow a periodic orbit and examining the steady-state solution that arises. After a sufficient amount of time, the various quantities (energy and angular momentum fluxes through the horizon and out to infinity, the self-force, etc.) should converge to the frequency domain results. The frequency domain codes produce extremely high-accuracy results, providing excellent comparison data for validating other codes in the EMRI community.

In the time domain, a wide array of codes have been developed, taking advantage of several approximations to numerically evolve these systems. Many of these methods fall under the broad class of approximations referred to as the *adiabatic approximation*, which assumes that secular self-force effects act on a timescale much longer than the orbital period. While this is valid early in the inspiral, it fails to capture the physics during the plunge, when the particle trajectory transitions from a path well-approximated by geodesic orbits to a quasi-radial infall into the central body.

One specific choice in this class is the radiative approximation. Work by Gal'tsov—and expanded by Mino—showed that the averaged rates of change in the orbital parameters caused by the true self-force could be reproduced by choosing a specific radiative Green's function to construct the self-force [24, 25]. This approximation therefore captured all the dissipative parts of the self-force, but (by construction) ignored the conservative part; this has been successfully implemented in numerical schemes already [26–28]. The decomposition into conservative and dissipative forces occurs in most physical problems, though they may not be discussed explicitly. Dissipative forces (such as friction) depend on the path of the

particle, and their existence is often the reason a numerical approach is required. They usually 'bleed' energy and momentum out of the system in some way, converting it into heat, gravitational waves, etc. Conservative forces are path-independent, but they still have effects on the evolution. To see how ignoring the conservative piece of the self-force affects the results, Hinderer and Flanagan [29] performed a comprehensive study of these issues, and they explicitly split the self-force into dissipative and conservative parts, by order:

$$f^{\mu} = \frac{m}{M}(f^{\mu}_{(1)d} + f^{\mu}_{(1)c}) + \frac{m^2}{M^2}(f^{\mu}_{(2)d} + f^{\mu}_{(2)c}) + \cdots \tag{3.11}$$

Their primary result was to express the orbital phase

$$\phi = \frac{M^2}{m}\left(\phi^{(0)} + \frac{m}{M}\phi^{(1)} + \cdots\right) \tag{3.12}$$

in terms of the quantities in Equation 3.11. They found that $\phi^{(0)}$ depends on the averaged part of $f^{\mu}_{(1)d}$ and $\phi^{(1)}$ depends on $f^{\mu}_{(1)c}$, the oscillatory part of $f^{\mu}_{(1)d}$, and the averaged part of $f^{\mu}_{(2)d}$. This matches analysis of numerical results which show that the radiative approach introduces cumulative errors in both the orbital phase and gravitational wave itself [30–32]. Therefore, the radiative approximation, and most other adiabatic approximations, fail to capture the first order correction to the orbital phase and is insufficient for parameter estimation for LISA, though it may be sufficient for detection. Clearly, a method which incorporates the conservative piece of the self-force is needed.

3.5. Self-Consistent Evolution

Ideally, a code should be developed which uses the full instantaneous self-force to drive the particle's motion, as this would provide the most realistic evolution for the particle. In addition, it would provide insight into the accuracy of the other evolution schemes. The self-consistent approach aims to do exactly this by driving the evolution with the calculated self-force at each timestep. Thus far, however, there is only one concerted effort to produce such a code. The code written by Peter Diener is designed to do precisely this [3]. The

original code could only simulate the self-force with a scalar charge in the Schwarzschild spacetime. In addition, the code has instability issues at long runtimes. Still, significant progress has been made towards producing a functional self-consistent evolution code, and it is included as part of the Einstein Toolkit as a standalone code (i.e. a code not using the Cactus Framework). The code is also part of the Black Hole Perturbation Toolkit. Much of my research has focused on expanding this code into the far more difficult gravitational case, but I also contributed to the development of the scalar self-force code through my derivation of the expressions for the angular momentum fluxes at the horizon and $\mathscr{I}^+$ [33].

3.6. Effective Source Approach

One major difficulty with performing a self-consistent evolution is the computational expense. Determining the field using mode-sum decomposition is expensive, and the additional cost of computing the self-force for every iteration for all the modes is prohibitively expensive. As such, the effective source approach—also known as the puncture approach, in reference to a similar strategy employed in standard numerical relativity—was developed to regularize the source itself instead of performing the regularization during the mode-sum [34–39]. This method is used in other codes, but it is critically important for designing an efficient self-consistent evolution code.

To demonstrate this approach, I shall again consider a scalar charge q with retarded potential Φ_{ret}. The coordinate x parameterizes the spacetime and represents the location of interest within the field; the coordinate $z(\tau)$ represents the particle's parameterized worldline γ. The variable r measures the distance from x to the worldline. The source is a delta function source along the particle's worldline. For simplicity, I define the functional

$$\sigma(x,z) \equiv -4\pi \int_\gamma \frac{\delta^{(4)}\left(x - z(\tau)\right)}{\sqrt{-g}} d\tau. \tag{3.13}$$

The wave equation for the retarded potential Φ_{ret} is

$$\Box \Phi_{ret} = q\sigma(x,z). \tag{3.14}$$

The singular field Φ_S obeys the same equation. I now introduce an approximate singular field $\tilde{\Phi}_S$ which obeys the equation

$$\Box\tilde{\Phi}_S = q\sigma(x,z) + \mathcal{O}(r^n) \tag{3.15}$$

for some integer $n \geq 0$. The approximate regular field is then

$$\tilde{\Phi}_R = \Phi_{ret} - W\tilde{\Phi}_S \tag{3.16}$$

where W is a window function with specifically chosen attributes. The wave equation for the regular field is then

$$\Box\tilde{\Phi}_R = \Box\Phi_{ret} - \Box\left(W\tilde{\Phi}_S\right) := S(x,z) \tag{3.17}$$

where $S(x,z)$ is the effective source term. Examining the source term,

$$\begin{aligned} S(x,z) &= \Box\Phi_{ret} - \Box\left(W\tilde{\Phi}_S\right) \\ &= q\sigma(x,z) - W\Box\tilde{\Phi}_S - \tilde{\Phi}_S\Box W - 2\nabla_\alpha W\nabla^\alpha\tilde{\Phi}_S \end{aligned} \tag{3.18}$$

In the limit that x approaches the particle's worldline ($x \to z$), the two leftmost terms become $q\sigma(x,z) - W\left(q\sigma(x,z) + \mathcal{O}(r^n)\right)$. If the window function W is chosen to approach unity in the limit $x \to z$, then the two singular terms will cancel out exactly. The other two terms each introduce another restriction on W to enforce a finite effective source. For the third term, $\Phi_S \propto r^{-1}$, which requires that $\Box W \propto r^p, p \geq 1$. For the fourth term, $\nabla\Phi_S \propto r^{-2}$, which requires that $\nabla W \propto r^q, q \geq 2$. These two relations mean that $W \propto r^i, i \geq 3$. To preserve the symmetry of the source around the particle, i should be even, meaning that $W \propto r^{4+2m}, m \geq 0$. The value of m is chosen such that the error introduced by the window function is of the same order as the error of the approximate singular field. All of these restrictions have the added benefit of directly relating the effective regular field to the self-

force:

$$\begin{aligned}\lim_{x\to z}\nabla_\alpha\tilde{\Phi}_R &= \lim_{x\to z}\left(\nabla_\alpha\Phi_{ret} - W\nabla_\alpha\tilde{\Phi}_S\right) - \lim_{x\to z}\tilde{\Phi}_S\nabla_\alpha W \\ &= \lim_{x\to z}\left(\nabla_\alpha\Phi_{ret} - \nabla_\alpha\tilde{\Phi}_S\right) \\ &= q^{-1}F_\alpha\end{aligned} \tag{3.19}$$

The effective-source method reduces the singular nature of Equation 3.14 to the equation

$$\Box\tilde{\Phi}_R = S(x,z) \tag{3.20}$$

which consists entirely of regular functions. The window function is also chosen to only have support in the immediate vicinity of the particle's worldline. This ensures that the effective source is not smeared out across the space, and that the numerically calculated field $\tilde{\Phi}_R$ equals the true regular field at the particle but also equals the retarded field $\tilde{\Phi}_{ret}$ outside the region of support for W. Because of this, the regular effective field can be used for evolution throughout the entire domain.

3.7. Hyperboloidal Coordinates

Another method used to improve the evolution of the self-consistent code is hyperboloidal foliation [40]. The waveform needed for comparison with experiment must effectively be at future null infinity, which is usually denoted as $\mathscr{I}^+$. Due to computational limitations, the numerical domain cannot be extended far enough to approximate the result at $\mathscr{I}^+$, so some form of extrapolation technique is traditionally used. However, this requires a choice of boundary conditions for the outer boundary, which is not obvious for the EMRI system. Instead, the spatial coordinate can be compactified to compress the entire spatial coordinate into a finite coordinate space. First, I use the commonly used tortoise coordinate instead of the Schwarzschild radial coordinate. For completeness, the relation between the

Schwarzschild coordinate r and the tortoise coordinate r_* is

$$r_* = r + 2M\ln\left(\frac{r}{2M} - 1\right)$$

This coordinate transformation moves the horizon (at $r = 2M$) to $r_* = -\infty$. To compactify the radial coordinate, the tortoise coordinate is transformed to the compactified coordinate ρ:

$$r_* = \frac{\rho}{\Omega(\rho)} \tag{3.21}$$

The function Ω is non-negative over the entire compactified region and satisfies the conditions

$$\Omega(S) = 0,\ \Omega'(S) \neq 0$$

where S is the coordinate value corresponding to $\mathscr{I}^+$. Finally, the following are required to ensure continuity and differentiability along the transition interface from the tortoise coordinate region to the compactified region:

$$\begin{gathered}\Omega = 1\ \forall\ \rho \leq R_* \\ \frac{d^k\Omega}{d\rho^k}|_{\rho=R_*} = 0 \text{ for } 1 \leq k \leq k_{max}\end{gathered} \tag{3.22}$$

In Equations 3.22, R_* is the coordinate where $r_* = \rho$, and k_{max} determines the level of differentiability of the layer. These restrictions on Ω allow for a smooth transition of the fields across the transition layer between the two coordinate regions.

However, this compactification is not sufficient to resolve the waveform. After all, the coordinates in the grid going out along the spatial coordinate ρ will eventually contain more and more oscillations between each gridpoint. This means that the resolution of the grid is greatly reduced in the outer gridpoints, ruining the data from the better resolved tortoise region [41]. To resolve this, the Cauchy surface is chosen to asymptotically approach $\mathscr{I}^+$

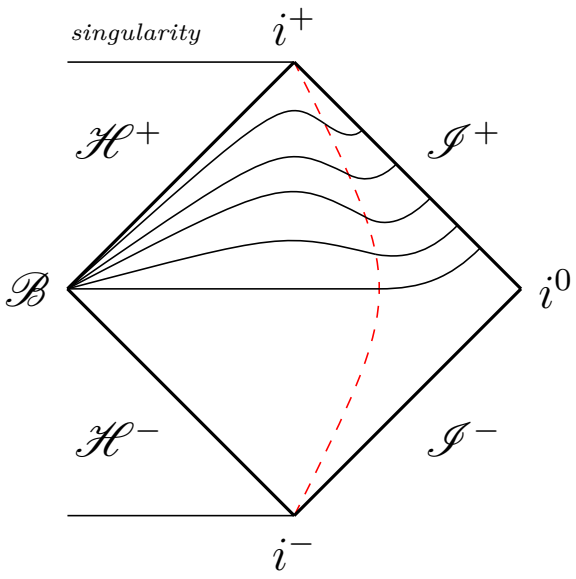

Figure 3.4. Penrose diagram of the hyperboloidal layer. The dashed line represents the interface where the coordinates transition from tortoise coordinates to hyperboloidal coordinates. Instead of the coordinates approaching spatial infinity i^0, the coordinates asymptote to null future infinity $\mathscr{I}^+$.

instead of spatial infinity in the outer region, which preserves the resolution [42]. This also resolves the outer boundary problem, as the ingoing characteristic speed at null infinity (and the outer coordinate) is zero, meaning nothing can propagate inwards. Therefore, the outer boundary condition is simply to do nothing.

To construct the hyperboloidal layer, several desirable traits should be established.

1. The timelike Killing vector field is invariant in the layer. This preserves the direction of time, again making the extraction of physical result simpler.

2. Outgoing null rays are invariant in the layer. This preserves the surface along which waveforms are traditionally plotted, making their extraction from the numerical code more straightforward.

3. The hyperboloidal coordinates $\{\tau, \rho\}$ must agree with the tortoise coordinates $\{t, r_*\}$ at the interface. This ensures that the coordinates are continuous, which will help with numerical implementation and stability.

For the following, I consider the transformation in the Schwarzschild spacetime. The first condition effectively requires that $\partial_t = \partial_\tau$, which means the transformation equation must be of the form

$$\tau = t - h(r_*) \tag{3.23}$$

In general, the height function h could depend on angular components, but not for a spherically symmetric space. For the second condition, invariant null rays is equivalent to

$$t - r_* = \tau - \rho \tag{3.24}$$

Combining Equations 3.23 and 3.24 gives the following for the height function:

$$h(r_*) = r_* - \rho(r_*) \tag{3.25}$$

Taking the derivative,

$$H \equiv \frac{dh}{dr_*} = 1 - \frac{d\rho}{dr_*} \tag{3.26}$$

The so-called boost function H is directly related to the Jacobian of the spatial compactification. By using these coordinates in the outer region, there is no need to extrapolate the waveform to null infinity, and no outer boundary conditions are required.

While this discussion focused on the outer boundary, I also perform a similar transformation at the inner boundary. As previously mentioned, the tortoise coordinate r_* takes the spatial coordinate r and stretches the domain $2M \leq r \leq \infty$ to $-\infty \leq r_* \leq \infty$. As such, an inner hyperboloidal layer can be added in the inner region as well. The only change from the above is in condition two. Instead of preserving the outgoing null rays, I preserve the ingoing null rays. This changes Equation 3.24 to be

$$t + r_* = \tau + \rho \tag{3.27}$$

which means that

$$h(r_*) = -(r_* + \rho(r_*)) \tag{3.28}$$

Taking the derivative,

$$H = -(1 + \frac{d\rho}{dr_*}) \tag{3.29}$$

In summary, I use three coordinate layers. The tortoise coordinates $\{t, r_*\}$ are used around the particle. At transition layer $r_* = \rho(r_*) = R_+$, I transform into the outer hyperboloidal coordinates $\{\tau, \rho\}$ defined by the first set of coordinate transformations. These coordinates end at $\rho_{max} = S_+$, which is equivalent to null infinity. At transition layer $r_* = \rho(r_*) = R_-$, I transform into the inner hyperboloidal coordinates $\{\tau, \rho\}$ defined by the second set of coordinate transformations. These coordinates end at $\rho_{min} = S_-$, which is the location of the horizon. Using these coordinates, I can avoid boundary conditions on both ends of the computational domain with no reduction in numerical resolution, as well as avoiding the need for extrapolation of the waveform.

3.8. Discontinuous Galerkin Method

3.8.1. Motivations

The final trick used to improve the numerical stability and reduce computational cost for these simulations is the Discontinuous Galerkin (DG) method. As a finite-element method, it reduces partial differential equations into a set of ordinary differential equations which are then solved using a standard numerical algorithm. I use the 4th order Runge-Kutta method, but the time integrator is chosen a runtime parameter and can be easily changed to any other implemented method.

To explain why the DG method is used, I first review the benefits and downsides of other common methods. For a more in-depth discussion of the DG method and its relation to other numerical methods, the book by Hesthaven and Warburton [43] serves as an excellent resource. Much of this section is a summary of the first few chapters of this book. I restrict my discussion to a single dimension, as the tensor spherical harmonics I use reduce the system

to a one-dimensional problem. For a simple example, consider the scalar conservation law in the space Ω with some solution $u_E(x,t)$, flux $f_E(u)$, and forcing function $g(x,t)$

$$\partial_t u_E + \partial_x f_E = g \tag{3.30}$$

along with appropriate initial and boundary conditions. To evolve this, the exact function u_E and f_E must be approximated with some u and f. The specifics of this approximation defines the method.

Finite difference methods (FDM) are the earliest methods developed, and they have the benefit of being very straightforward to implement. FDM involve dividing the space Ω with K gridpoints x^k and approximating the spatial derivative as

$$\partial_x f_E \approx \frac{f(x^{k+1},t) - f(x^{k-1},t)}{h^k - h^{k-1}} \tag{3.31}$$

where $h^k = x^{k+1} - x^k$ is the local grid spacing. The method is defined by requiring that the approximate solution u exactly matches u_E at each gridpoint. By doing this for every gridpoint, the system can be evolved forward using a time integration method. Extending FDM to higher orders is also relatively straightforward, as it amounts to extending the stencil used for approximating the spatial derivative from the nearest neighboring gridpoints to the next-nearest neighbors, and so on. Unfortunately, this class of methods also has several drawbacks. The source of this simplicity is the assumption that the grid is uniform, meaning that the resolution is fixed across the domain. This is a poor choice for most astrophysical systems, as the mesh needs to be far finer around sources than in empty space. This issue was mentioned in Chapter 2, and Carpet solves it by using adaptive mesh refinement. In that approach, each grid is uniform, but new grids are introduced in areas needing finer meshes. These methods also have difficulties handling sudden boundary layers (discontinuous material layers, star surfaces, etc.) and are prone to numerical instabilities when used for these applications.

To more naturally handle these types of systems, the requirement of uniform grids must be relaxed. This led to the development of finite volume methods (FVM) and finite element methods (FEM). FVM discretize Ω with K elements $D^k = \left[x^{k-\frac{1}{2}}, x^{k+\frac{1}{2}}\right]$. The approximate solution u^k is defined at x^k, the center of each cell. In this case, the approximation requires that the average over the cell matches the exact solution, and leads to the equation

$$h^k \partial_t u^k + f^{k+\frac{1}{2}} - f^{k-\frac{1}{2}} = h^k g^k \tag{3.32}$$

where the flux term has been changed to surface terms using the divergence theorem. Since the equation for u^k only depends on h^k, this method depends purely on the local geometry and does not impose conditions on the grid structure. Unfortunately, the fluxes are needed at the boundaries of the elements, while the solution is only known at the cell center. The flux reconstruction is non-trivial, but the simplest choice is just to average the cell centers to find the value at the interface

$$u^{k+\frac{1}{2}} = \frac{u^{k+1} + u^k}{2} \tag{3.33}$$

Then,

$$f^{k+\frac{1}{2}} = f\left(u^{k+\frac{1}{2}}\right) \tag{3.34}$$

For linear problems and uniform grids, this method is identical to FDM, but it has far more freedom in its allowed grid structure. However, extension to higher orders extends this stencil like the FDM, which once again places restrictions on the grid structure.

The source of this issue is the reliance on cell averages. To avoid this, let the elements instead be defined as $D^k = \left[x^{k-1}, x^{k+1}\right]$ with K elements and $K+1$ gridpoints. Within each element, the solution is approximated with

$$u(x) = \sum_{n=1}^{N_p} b_n \psi_n(x) \tag{3.35}$$

In the simple case of linear functions,

$$u(x) = \sum_{i=0}^{1} u(x^{k+i}) \ell_i^k(x) \tag{3.36}$$

where $\ell_i^k(x)$ is the linear Lagrange polynomial. Since each neighboring element shares a node at the boundary, the global representation of the solution

$$u = \sum_{k=1}^{K} u^k N^k(x) \tag{3.37}$$

can be constructed where $N^i(x^j) = \delta_{ij}$ form a basis. From this, one defines a space of test functions $\phi \in V$ and enforce that

$$\int_\Omega \left(\partial_t u + \partial_x f - g\right) \phi(x) dx = 0 \tag{3.38}$$

Different choices of test functions generate different numerical schemes. The Galerkin scheme chooses

$$\phi(x) \sum_{k=1}^{K} v(x^k) N^k(x) \tag{3.39}$$

which leads to the matrix representation

$$\boldsymbol{M} \partial_t \mathbf{u} + S\mathbf{f} = \boldsymbol{M}\mathbf{g} \tag{3.40}$$

where

$$M_{ij} = \int_\Omega N^i(x) N^j(x) dx \tag{3.41}$$

$$S_{ij} = \int_\Omega N^i(x) \partial_x N^j(x) dx \tag{3.42}$$

The vectors $\mathbf{u}$, $\mathbf{f}$, and $\mathbf{g}$ contain the values of these functions at the $K+1$ gridpoints. This is the basis of the FEM. There are no restrictions on the grid structure, and the higher

order methods are obtained by increasing the degrees of freedom within the element (see Equation 3.37), which allows for two different methods of increasing the accuracy. This so-called *hp*-adaptivity allows for the ability to change the local order of the method (*p*-adaptive) and the ability to change the local grid spacing (*h*-adaptive).

Of course, the FEM methods have their own drawbacks. By the form of Equation 3.40, this is an implicit method which requires inverting M, which can cause problems for time-dependent problems. Also, the symmetric basis functions N^i can cause stability issues with problems for which there is a specific directional flow of information (e.g. wave equations, advection equations, conservation laws). FDM and FVM resolve this through implementing upwinding, which biases the numerical method towards the 'upwind' direction to stabilize the simulation [44].

A desired numerical method would preserve the *hp*-adaptivity of FEM but without the global statement of basis in Equation 3.37 and need for an implicit evolution scheme. The Discontinuous Galerkin finite element method combines features from FVM and FEM to produce such a method. This method starts with the FEM discretization $D^k = \left[x^{k-1}, x^{k+1}\right]$ with K elements and $K+1$ gridpoints. However, the unknowns in the equation u^k are duplicated at the endpoints. For simplicity I consider the linear approximation (i.e. the solution within each element is approximated by a linear polynomial). Then, the solution vector $\mathbf{u}$ would take the form

$$\mathbf{u} = [u^1, u^2, u^2, u^3, \ldots, u^K, u^K, u^{K+1}] \tag{3.43}$$

and has length $2K$ instead of $K+1$. As before, the local solution in each element is expanded using Lagrange polynomials which are also used as the space of test functions. These functions are only defined on the element, so there are no smoothness restrictions between elements. Now, Equation 3.38 is instead only an integral over the local element

instead of the entire domain

$$\int_{D^k} \left(\partial_t u^k + \partial_x f^k - g\right) \ell_j^k(x) dx = 0 \tag{3.44}$$

This is possible only because of the duplicated points at the element boundaries. By taking inspiration from FVM, the divergence theorem can be used to give

$$\int_{D^k} \left(\ell_j^k(x)\partial_t u^k - f^k \partial_x \ell_j^k(x) - g\ell_j^k(x)\right) dx = -f^k \ell_j^k(x)|_{x^k}^{x^{k+1}} \tag{3.45}$$

In the case where $\ell_j^k(x)$ is constant, this reduces to Equation 3.32 for FVM. The surface terms in both equations serve the role of connecting the physics between the two elements. While there are many different prescriptions for calculating these surface terms, most use some form of numerical fluxes between the elements to account for these terms. From here, an equation similar to Equation 3.40 arises, except that the matrices are all local. The method is therefore explicit instead of implicit, and the matrix inversion is far less costly.

3.8.2. Implementation

The DG method combines features of FVM and FEM to produce a hybrid scheme which deals with localized approximations and non-uniform grids. These features all benefit the EMRI system, as the resolution must be higher near the particle than the rest of the space. The localization of the expressions also improves parallelism, an important feature for modern computational codes. The DG formalism can be expressed in either a modal or nodal form

$$u^k(x,t) = \sum_{n=1}^{N_p} \hat{u}(t)\psi_n(x) \tag{3.46}$$

$$= \sum_{i=1}^{N_p} u^k(x,t)\ell_i^k(x) \tag{3.47}$$

where $N = N_p - 1$ is the order of the interpolating polynomial. As before, $\ell_i^k(x)$ are the Lagrange polynomials, and $u^k(x,t)$ is the nodal representation. The modal representation is $\hat{u}(t)$, and the natural choice for the basis is

$$\psi_n = \tilde{P}_{n-1} = a_n P_{n-1} \tag{3.48}$$

$$a_n = \left(\sqrt{\frac{2n}{2n+1} - 1} \right)^{-1} \tag{3.49}$$

where P_{n-1} are the Legendre polynomials and a_n are normalization coefficients. These $\tilde{P}_{n-1}$ are from the class of Jacobi polynomials (also called hypergeometric polynomials) which have the form $P_n^{(\alpha,\beta)}(x)$ with $\alpha = \beta = 0$.

While our code uses the nodal representation, I introduce both so as to properly discuss the choice of grid points within each element. To find the $\hat{u}$, the functions u must be projected onto the $\tilde{P}$ basis, which means computing the inner product

$$\hat{u} = (u, \tilde{P}_{n-1}) \tag{3.50}$$

This could be approximated by calculating the Gaussian quadrature over the N_p gridpoints in the element. Alternatively, we can enforce that the modal representation exactly matches the nodal representation at each interpolation point ξ_i

$$u(\xi_i, t) = \sum_{n=1}^{N_p} \hat{u}(t) \tilde{P}_{n-1}(\xi_i) \tag{3.51}$$

which can be concisely written as

$$\boldsymbol{V}\tilde{\mathbf{u}} = \mathbf{u} \tag{3.52}$$

where $\boldsymbol{V}$ is the generalized Vandermonde matrix with elements

$$V_{ij} = \tilde{P}_{j-1}(\xi_i) \tag{3.53}$$

Now that the basis is well-defined, the next step is to choose the gridpoints ξ_i, as nothing has yet restricted our choice of interpolating points. The simplest is obviously uniform spacing, but to make a good choice requires some criterion for establishing the 'goodness' of a set of ξ_i. To do this, I consider the maximum norm

$$\| u_E - u \| = \| u_E - u_* + u_* - u \|$$

where u_* is the best approximating polynomial for order N. By the triangle inequality,

$$\| u_E - u \| \leq \| u_E - u_* \| + \| u_* - u \|$$

However, u is a projection of u_E. The function u_* is in the projected space, but it is also made up of the polynomials, so it is a projection of itself. If the projection operator is given by F, then

$$\begin{aligned} \| u_* - u \| &= \| F(u_*) - F(u_E) \| \\ &= \| F \| \| u_* - u_E \| \end{aligned}$$

which is a special case of Lebesgue's lemma. This gives

$$\| u_E - u \| \leq (1 + \Lambda) \| u_E - u_* \|$$

where $\Lambda = \| F \|$. it follows that the ideal polynomial representation arises when Λ is minimized.

In the modal basis, u is given by Equation 3.51. However, we have restricted the interpolation to be exact at the gridpoints. Then, for the purposes of computing the norm it is equally valid to use Equation 3.46. Then, the norm of the projection is given by the

Lebesgue constant

$$\Lambda = \max_r \sum_{i=1}^{N_p} |\ell_i(\xi_r)| \tag{3.54}$$

However, this criterion for optimizing the grid is in terms of the nodal basis. To extract information about the local grid structure from this result requires returning to Equation 3.52. By uniqueness of both of the polynomial interpolations, it follows from Equation 3.52 that

$$\boldsymbol{V}^T \ell(r) = \tilde{\mathbf{P}}(r) \tag{3.55}$$

where $\ell(r)$ and $\tilde{\mathbf{P}}(r)$ are vectors of length N containing the polynomials for the respective basis. To minimize Λ, the particular solution of $\ell(r)$ should be minimized. The solution to the linear system of equations in Equation 3.55 can be found using Cramer's rule. For a general equation of the form

$$\boldsymbol{A}\mathbf{x} = \mathbf{b} \tag{3.56}$$

the solution is

$$x_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})} \tag{3.57}$$

where $\boldsymbol{A}_i$ replaces the i-th column of $\boldsymbol{A}$ with $\mathbf{b}$. In the case of Equation 3.55, the denominator is simply the transposed Vandermonde matrix $\boldsymbol{V}^T$. Thus, the desired grid spacing maximizes the determinant of $\boldsymbol{V}$, which is defined in terms of the modal basis. In one dimension, the optimized gridpoint locations are known as the Legendre-Gauss-Lobatto (LGL) quadrature points [45, 46].

Our code uses the nodal representation, and the Jacobi polynomials, Vandermonde matrix, integration weights for the LGL quadrature points, etc. are all implemented in the `Grid` submodule. In practice, the LGL points congregate near the element boundaries. This is unsurprising, as capturing the effects of the fluxes at the element boundaries should require higher resolution than the center of the element domain. By using the DG method, our code is *hp*-adaptive, as the order of the (p)olynomial interpolation within each element and the

element size (h) can be scaled independently. Also, the fluxes at the element boundaries only depend on the elements which share that boundary, unlike FVM. The method is also fully explicit in time, unlike FEM. In summary, the DG method allows for varied grid structures which are well-suited to the EMRI self-force problem and have many benefits over other traditional methods.

Chapter 4. Self-force Implementation in the Regge-Wheeler-Zerilli Gauge

4.1. Tensor Spherical Harmonics

In the following two chapters, I explain two different EMRI codes I have developed to simulate the gravitational self-force. The work in this chapter uses the Regge-Wheeler-Zerilli (RWZ) gauge, and the following chapter uses the Lorenz gauge. The RWZ gauge is defined to simplify the results from the tensor spherical harmonic decomposition of the spacetime metric. As such, the exact specification of the gauge is significantly more complicated than the Lorenz or other more common gauges. This gauge is used because the resulting evolution equations are significantly easier to deal with than other gauges. This methodology has reached significant maturity and is presented in it's entirety in a covariant, gauge-invariant formalism by Martel and Poisson [47]. Here, I shall summarize the derivation of the RWZ gauge and discuss its benefits, adopting similar notation to Martel and Poisson.

To begin, I take the space-time manifold M^4 and split it into two submanifolds, M^2 and S^2. The lowercase letters $(a, b, c, \dots)$ are used for indices in the M^2 submanifold, and the uppercase letters $(A, B, C, \dots)$ are used for indices in the S^2 submanifold. In Schwarzschild coordinates, the submanifold M^2 represents the (t, r) plane, and S^2 represents the (θ, ϕ) two-sphere. I also define coordinates, metrics, and covariant derivatives for these submanifolds. The coordinates x^a span M^2, and θ^A spans S^2. ∇_a is the covariant derivative associated with M^2, and D_A is the covariant derivative associated with S^2. The 2-metrics are given by g_{ab} and Ω_{AB}, respectively. The indices a, b, c, etc. run over the values 0 and 1, while the indices A, B, C, etc. run over the values 2 and 3. Then, the full space-time metric can be given in terms of the submanifold quantities:

$$ds^2 = g_{ab}dx^a dx^b + r^2\Omega_{AB}d\theta^A d\theta^B$$

To express the perturbation equations in a general form, I also require several other terms. First, I define the dual vector

$$r_a \equiv \frac{\partial r}{\partial x^a}$$

which is normal to constant-r surfaces (r being the Schwarzschild radial coordinate). This also provides an alternative form for the frequently seen factor

$$f \equiv 1 - \frac{2M}{r} = r^a r_a$$

Next is the Levi-Cevita tensor for the M^2 submanifold ϵ_{ab}. Finally,

$$t^a \equiv -\epsilon^{ab} r_b$$

is the time-like Killing vector. With this, I can begin the derivation of the RWZ gauge.

The RWZ derivation is generally split into the even and odd sectors. The even tensor spherical harmonics are designated by a Y, and the odd tensor spherical harmonics are designated by a X. The tensor spherical harmonics are further divided by their indices: scalar, vector, and tensorial. The scalar spherical harmonics are simply the usual $Y^{\ell m}$. The vector spherical harmonics are

$$Y^{\ell m}_A = D_A Y^{\ell m} \tag{4.1}$$

$$X^{\ell m}_A = -\epsilon^B_A D_B Y^{\ell m} \tag{4.2}$$

Finally, the tensorial spherical harmonics are

$$Y^{\ell m}_{AB} = \left[D_A D_B + \frac{1}{2}\ell(\ell+1)\Omega_{AB} \right] Y^{\ell m} \tag{4.3}$$

$$X^{\ell m}_{AB} = -\frac{1}{2}\left[\epsilon^C_A D_B + \epsilon^C_B D_A \right] D_C Y^{\ell m} \tag{4.4}$$

I can now decompose the perturbed metric $\tilde{g}_{\alpha\beta}$ using the tensor spherical harmonics and the metrics on the submanifolds

$$\begin{aligned} \tilde{g}_{ab} &= g_{ab} + h_{ab} \\ \tilde{g}_{aB} &= h_{aB} \\ \tilde{g}_{AB} &= r^2\Omega_{AB} + h_{AB} \end{aligned} \tag{4.5}$$

where $h_{\alpha\beta}$ is the metric perturbation. As a reminder of notation, there are several different covariant derivatives and metrics. The full perturbed metric is $\tilde{g}_{\alpha\beta}$. The background 4-metric is $\overline{g}_{\alpha\beta}$, and the associated covariant derivative is $\overline{\nabla}_a$. On the submanifold M^2, the metric is g_{ab} and the associated derivative is ∇_a. On the submanifold S^2, the metric is Ω_{AB} and the associated derivative is D_A. The last tool we need is an understanding of how the metric perturbation behaves in each submanifold. I begin by defining the dual vector

$$\Xi_\alpha \equiv (\Xi_a, \Xi_A)$$

along with its tensor spherical harmonic decomposition

$$\Xi_a \equiv \begin{cases} \sum_{\ell m} \xi_a^{\ell m} Y^{\ell m}, & \text{even parity} \\ 0, & \text{odd parity} \end{cases}$$

$$\Xi_A \equiv \begin{cases} \sum_{\ell m} \xi^{\ell m} Y_A^{\ell m}, & \text{even parity} \\ \sum_{\ell m} \xi^{\ell m} X_A^{\ell m}, & \text{odd parity} \end{cases}$$

where $\xi_a^{\ell m}$ and $\xi^{\ell m}$ depend only on the coordinates x^a. I use this to generate a gauge transformation of the form

$$\overline{\nabla}_\alpha \overline{g}_{\beta\gamma} = \overline{\nabla}_\alpha \overline{g}'_{\beta\gamma} \equiv \overline{\nabla}_\alpha \left(\overline{g}_{\beta\gamma} - \overline{\nabla}_\alpha \Xi_\beta - \overline{\nabla}_\beta \Xi_\alpha \right) = 0$$

Since gauge transformations are generally assumed to be 'small', these new factors can be absorbed purely into the metric perturbation, leading to the gauge transformation

$$h'_{\alpha\beta} \equiv h_{\alpha\beta} - \overline{\nabla}_\alpha \Xi_\beta - \overline{\nabla}_\beta \Xi_\alpha$$

For this expression to be useful, it must be expressed with quantities defined on the submanifolds M^2 and S^2. While the general form will be the same as above, additional terms will arise from the Christoffel symbols of the 4-metric. I must first express the Christoffel symbols of $\overline{g}$ in terms of the quantities in the submanifolds. The symbols can be split into those with all indices in a single submanifold and those with mixed symbols. Starting with the M^2 symbols,

$$\begin{aligned}
\Gamma^a_{bc} &= \frac{1}{2}\overline{g}^{a\gamma}\left(\partial_b \overline{g}_{\gamma c} + \partial_c \overline{g}_{\gamma b} - \partial_\gamma \overline{g}_{bc}\right) \\
&= {}^M\Gamma^a_{bc} + \frac{1}{2}\overline{g}^{aD}\left(\partial_b \overline{g}_{Dc} + \partial_c \overline{g}_{Db} - \partial_D \overline{g}_{bc}\right) \\
&= {}^M\Gamma^a_{bc}
\end{aligned}$$

where ${}^M\Gamma^a_{bc}$ is the Christoffel symbol for the submanifold M^2. In the final step I use the fact that the mixed terms $\overline{g}_{aB} = 0$. Similarly,

$$\begin{aligned}
\Gamma^A_{BC} &= \frac{1}{2}\overline{g}^{A\gamma}\left(\partial_B \overline{g}_{\gamma C} + \partial_C \overline{g}_{\gamma B} - \partial_\gamma \overline{g}_{BC}\right) \\
&= {}^S\Gamma^A_{BC}
\end{aligned}$$

The mixed Christoffel symbols are Γ^A_{bc}, Γ^a_{BC}, Γ^A_{Bc}, and Γ^a_{Bc}. First,

$$\begin{aligned}
\Gamma^A_{bc} &= \frac{1}{2}\overline{g}^{A\gamma}\left(\partial_b\overline{g}_{\gamma c} + \partial_c\overline{g}_{\gamma b} - \partial_\gamma\overline{g}_{bc}\right) \\
&= \frac{1}{2}\overline{g}^{AD}\left(\partial_b\overline{g}_{Dc} + \partial_c\overline{g}_{Db} - \partial_D\overline{g}_{bc}\right) \\
&= -\frac{1}{2}\overline{g}^{AD}\partial_D\overline{g}_{bc} \\
&= 0
\end{aligned}$$

The final step follows from the knowledge that $\overline{g}_{ab} = g_{ab}$ is independent of the angular components. Similarly,

$$\begin{aligned}
\Gamma^a_{BC} &= \frac{1}{2}\overline{g}^{a\gamma}\left(\partial_B\overline{g}_{\gamma C} + \partial_C\overline{g}_{\gamma B} - \partial_\gamma\overline{g}_{BC}\right) \\
&= \frac{1}{2}\overline{g}^{ad}\left(\partial_B\overline{g}_{dC} + \partial_C\overline{g}_{dB} - \partial_d\overline{g}_{BC}\right) \\
&= -\frac{1}{2}\overline{g}^{ad}\partial_d\overline{g}_{BC} \\
&= -\frac{1}{2}\overline{g}^{ad}\partial_d\left(r^2\Omega_{BC}\right) \\
&= -r\overline{g}^{ad}r_d\Omega_{BC} \\
&= -rr^a\Omega_{BC}
\end{aligned}$$

The third combination is

$$\begin{aligned}
\Gamma^A_{Bc} &= \frac{1}{2}\overline{g}^{A\gamma}\left(\partial_B\overline{g}_{\gamma c} + \partial_c\overline{g}_{\gamma B} - \partial_\gamma\overline{g}_{Bc}\right) \\
&= \frac{1}{2}\overline{g}^{AD}\left(\partial_B\overline{g}_{Dc} + \partial_c\overline{g}_{DB} - \partial_D\overline{g}_{Bc}\right) \\
&= \frac{1}{2}\overline{g}^{AD}\partial_c\overline{g}_{DB} \\
&= \frac{1}{2}\overline{g}^{AD}\partial_c\left(r^2\Omega_{DB}\right) \\
&= rr_c\overline{g}^{AD}\Omega_{DB} \\
&= r^{-1}r_c\Omega_{DB}\Omega_{DB} \\
&= r^{-1}r_c\delta^A_B
\end{aligned}$$

Finally,

$$\begin{aligned}
\Gamma^a_{Bc} &= \frac{1}{2}\overline{g}^{a\gamma}\left(\partial_B\overline{g}_{\gamma c} + \partial_c\overline{g}_{\gamma B} - \partial_\gamma\overline{g}_{Bc}\right) \\
&= \frac{1}{2}\overline{g}^{ad}\left(\partial_B\overline{g}_{dc} + \partial_c\overline{g}_{dB} - \partial_d\overline{g}_{Bc}\right) \\
&= 0
\end{aligned}$$

Summarizing, the non-zero Christoffel symbols are

$$\begin{aligned}
\Gamma^a_{bc} &= {}^M\Gamma^a_{bc} \\
\Gamma^A_{BC} &= {}^S\Gamma^A_{BC} \\
\Gamma^a_{BC} &= -rr^a\Omega_{BC} \\
\Gamma^A_{Bc} &= r^{-1}r_c\delta^A_B
\end{aligned}$$

As a reminder, the goal is to express the metric perturbation with a gauge transformation in terms of only quantities on the submanifolds. These Christoffel symbols are needed to

map the covariant derivative of the gauge transformation onto the submanifolds. On M^2,

$$\begin{aligned}\overline{\nabla}_a \Xi_b &= \partial_a \Xi_b - \Gamma^{\gamma}_{ab} \Xi_{\gamma} \\ &= \partial_a \Xi_b - \Gamma^{d}_{ab} \Xi_d - \Gamma^{D}_{ab} \Xi_D \\ &= \nabla_a \Xi_b\end{aligned}$$

In the mixed components, the angular derivative gives

$$\begin{aligned}\overline{\nabla}_A \Xi_b &= \partial_A \Xi_b - \Gamma^{\gamma}_{Ab} \Xi_{\gamma} \\ &= \partial_A \Xi_b - \Gamma^{d}_{Ab} \Xi_d - \Gamma^{D}_{Ab} \Xi_D \\ &= D_A \Xi_b - r^{-1} r_b \delta^D_A \Xi_D \\ &= D_A \Xi_b - r^{-1} r_b \Xi_A\end{aligned}$$

and the M^2 derivative gives

$$\begin{aligned}\overline{\nabla}_a \Xi_B &= \partial_a \Xi_B - \Gamma^{\gamma}_{aB} \Xi_{\gamma} \\ &= \partial_a \Xi_B - \Gamma^{d}_{aB} \Xi_d - \Gamma^{D}_{aB} \Xi_D \\ &= \nabla_a \Xi_B - r^{-1} r_a \delta^D_B \Xi_D \\ &= \nabla_a \Xi_B - r^{-1} r_a \Xi_B\end{aligned}$$

For both of the previous expressions, the reason the partial derivatives become covariant derivatives is because they are acting on objects which are tensors in a different space. For example, the covariant derivative D_A treats Ξ_b as a scalar, and so it does not generate any

Christoffel symbols. Finally, the S^2 component is

$$\begin{aligned}\overline{\nabla}_A \Xi_B &= \partial_A \Xi_B - \Gamma^{\gamma}_{AB} \Xi_{\gamma} \\ &= \partial_A \Xi_B - \Gamma^{d}_{AB} \Xi_{d} - \Gamma^{D}_{AB} \Xi_{D} \\ &= D_A \Xi_B + r r^d \Omega_{BC} \Xi_d\end{aligned}$$

These quantities finally allow for the expression of the gauge transformation purely in terms of quantities from the submanifolds. The M^2 component is simply

$$\begin{aligned}h'_{ab} &= h_{ab} - \overline{\nabla}_a \Xi_b - \overline{\nabla}_b \Xi_a \\ &= h_{ab} - \nabla_a \Xi_b - \nabla_b \Xi_a\end{aligned}$$

Unsurprisingly, the M^2 component has no additional factors, as only the angular nature of S^2 introduces a difference between the derivatives. For the mixed components,

$$\begin{aligned}h'_{aB} &= h_{aB} - \overline{\nabla}_a \Xi_B - \overline{\nabla}_B \Xi_a \\ &= h_{aB} - \nabla_a \Xi_B - D_B \Xi_a + \frac{2}{r} r_a \Xi_B\end{aligned}$$

Finally, for S^2,

$$\begin{aligned}h'_{AB} &= h_{AB} - \overline{\nabla}_A \Xi_B - \overline{\nabla}_B \Xi_A \\ &= h_{AB} - D_A \Xi_B - D_B \Xi_A - 2 r r^c \Xi_c \Omega_{AB}\end{aligned}$$

Summarizing, the rules for applying the gauge transformation on the metric perturbation are

$$h'_{ab} \equiv h_{ab} - \nabla_a \Xi_b - \nabla_b \Xi_a \tag{4.6}$$

$$h'_{aB} \equiv h_{aB} - \nabla_a \Xi_B - D_B \Xi_a + \frac{2}{r} r_a \Xi_B \tag{4.7}$$

$$h'_{AB} \equiv h_{AB} - D_A \Xi_B - D_B \Xi_A - 2rr^c \Xi_c \Omega_{AB} \tag{4.8}$$

To examine the mode-decomposed quantities, I also need the derivatives of Ξ in their spherical harmonic form. In the even sector

$$\nabla_a \Xi_b = \sum_{\ell m} Y^{\ell m} \nabla_a \xi_b^{\ell m} \tag{4.9}$$

$$\nabla_a \Xi_B = \sum_{\ell m} Y_B^{\ell m} \nabla_a \xi^{\ell m} \tag{4.10}$$

$$D_A \Xi_b = \sum_{\ell m} \xi_b^{\ell m} Y_A^{\ell m} \tag{4.11}$$

$$\begin{aligned} D_A \Xi_B &= \sum_{\ell m} \xi^{\ell m} D_A Y_B^{\ell m} \\ &= \sum_{\ell m} \xi^{\ell m} (Y_{AB}^{\ell m} - \frac{1}{2} \ell(\ell+1) \Omega_{AB} Y^{\ell m}) \end{aligned} \tag{4.12}$$

In the odd sector the non-zero derivatives are

$$\nabla_a \Xi_B = \sum_{\ell m} X_B^{\ell m} \nabla_a \xi^{\ell m} \tag{4.13}$$

$$D_B \Xi_A + D_A \Xi_B = \sum_{\ell m} \xi^{\ell m} \left(D_B X_A^{\ell m} + D_A X_B^{\ell m} \right) \tag{4.14}$$

$$= 2 \sum_{\ell m} \xi^{\ell m} X_{AB}^{\ell m} \tag{4.15}$$

where I have chosen to only consider the combination $D_B \Xi_A + D_A \Xi_B$. I do this because only the combined term is needed, and it takes on a simpler form than just the covariant derivative. The metric perturbation can now be expanded using the tensor spherical harmonics and

examined for each parity. For the following sections, summations over ℓ have the bounds $1 < \ell < \infty$, as $\ell = 0$ and $\ell = 1$ must be handled separately. Summations over m have the bounds $-\ell \leq m \leq \ell$.

4.1.1. Even Parity

For the even parity, the metric perturbation has three non-zero components:

$$h_{ab} = \sum_{\ell m} p_{ab}^{\ell m} Y^{\ell m} \tag{4.16}$$

$$h_{aB} = \sum_{\ell m} j_{a}^{\ell m} Y_{B}^{\ell m} \tag{4.17}$$

$$h_{AB} = r^2 \sum_{\ell m} \left(K^{\ell m} \Omega_{AB} Y^{\ell m} + G^{\ell m} Y_{AB}^{\ell m} \right) \tag{4.18}$$

The fields $p_{ab}^{\ell m}$, $j_{a}^{\ell m}$, $K^{\ell m}$, and $G^{\ell m}$ are all defined on M^2 and only depend on x^a. To derive the RWZ gauge, I use Equations 4.6–4.12 to derive the transformations for the scalar fields in Equations 4.16–4.18. Solving for the transformation for $p_{ab}^{\ell m}$,

$$\begin{aligned} h'_{ab} &= h_{ab} - \nabla_a \Xi_b - \nabla_b \Xi_a \\ &= \sum_{\ell m} p_{ab}^{\ell m} Y^{\ell m} - \sum_{\ell m} Y^{\ell m} \nabla_a \xi_b^{\ell m} - \sum_{\ell m} Y^{\ell m} \nabla_b \xi_a^{\ell m} \\ &= \sum_{\ell m} Y^{\ell m} \left(p_{ab}^{\ell m} - \nabla_a \xi_b^{\ell m} - \nabla_b \xi_a^{\ell m} \right) \end{aligned}$$

For $j_{a}^{\ell m}$,

$$\begin{aligned} h'_{aB} &= h_{aB} - \nabla_a \Xi_B - D_B \Xi_a + \frac{2}{r} r_a \Xi_B \\ &= \sum_{\ell m} j_a^{\ell m} Y_B^{\ell m} - \sum_{\ell m} Y_B^{\ell m} \nabla_a \xi^{\ell m} - \sum_{\ell m} \xi_a^{\ell m} Y_B^{\ell m} + \frac{2}{r} r_a \sum_{\ell m} Y_B^{\ell m} \xi^{\ell m} \\ &= \sum_{\ell m} Y_B^{\ell m} \left(j_a^{\ell m} - \nabla_a \xi^{\ell m} - \xi_a^{\ell m} + \frac{2}{r} r_a \xi^{\ell m} \right) \end{aligned}$$

Finally for K and G,

$$\begin{aligned}
h'_{AB} &= h_{AB} - D_A\Xi_B - D_B\Xi_A - 2rr^c\Xi_c\Omega_{AB} \\
&= r^2\sum_{\ell m}\left(K^{\ell m}\Omega_{AB}Y^{\ell m} + G^{\ell m}Y^{\ell m}_{AB}\right) - \sum_{\ell m}\xi^{\ell m}(Y^{\ell m}_{AB} - \frac{1}{2}\ell(\ell+1)\Omega_{AB}Y^{\ell m}) \\
&\qquad - \sum_{\ell m}\xi^{\ell m}(Y^{\ell m}_{AB} - \frac{1}{2}\ell(\ell+1)\Omega_{AB}Y^{\ell m}) + 2rr^c\Omega_{AB}\sum_{\ell m}Y^{\ell m}\xi^{\ell m}_c \\
&= \sum_{\ell m}Y^{\ell m}_{AB}\left(r^2G^{\ell m} - 2\xi^{\ell m}\right) + \sum_{\ell m}\Omega_{AB}Y^{\ell m}\left(r^2K^{\ell m} - \ell(\ell+1)\xi^{\ell m} + 2rr^c\xi^{\ell m}_c\right)
\end{aligned}$$

Since all the gauge transformations cleanly separate by ℓ and m modes, these equations give the transformations for the scalar fields in the even sector. Dropping the ℓ,m superscripts,

$$p'_{ab} = p_{ab} - \nabla_a\xi_b - \nabla_b\xi_a \tag{4.19}$$

$$j'_a = j_a - \nabla_a\xi - \xi_a + \frac{2}{r}r_a\xi \tag{4.20}$$

$$K' = K + \frac{\ell(\ell+1)}{r^2}\xi - \frac{2}{r}r^a\xi_a \tag{4.21}$$

$$G' = G - \frac{2}{r^2}\xi \tag{4.22}$$

Now, I use the gauge freedoms to reduce the number of scalar fields. To do so, I first need to find a gauge invariant quantity to decide how to define this new gauge. I start by examining K. First, notice that the quantity

$$\frac{\ell(\ell+1)}{r^2}\xi$$

can be canceled out with the term from

$$\frac{\ell(\ell+1)}{2}G'$$

Next, the term

$$-\frac{2}{r}r^a\xi_a$$

is obtained with the contraction

$$-\frac{2}{r}r^a j_a$$

Of course, simply adding these together will not perfectly cancel all the terms. Some other term needs to be added to make this construction gauge invariant. So far, the expression for some gauge-invariant quantity $\overline{K}$ is

$$\overline{K} = K + \frac{\ell(\ell+1)}{2}G - \frac{2}{r}r^a j_a + ?$$

Transforming this gives

$$\begin{aligned}
\overline{K} &= K' + \frac{\ell(\ell+1)}{2}G' - \frac{2}{r}r^a j_a' + ?' \\
&= K + \frac{\ell(\ell+1)}{r^2}\xi - \frac{2}{r}r^a\xi_a + \frac{\ell(\ell+1)}{2}\left(G - \frac{2}{r^2}\xi\right) - \frac{2}{r}r^a\left(j_a - \nabla_a\xi - \xi_a + \frac{2}{r}r_a\xi\right) + ?' \\
&= K + \frac{\ell(\ell+1)}{2}G - \frac{2}{r}r^a j_a - \frac{2}{r}r^a\left(-\nabla_a\xi + \frac{2}{r}r_a\xi\right) + ?'
\end{aligned}$$

From here, the solution is somewhat clear. The remaining term is just the covariant derivative

$$\nabla_a\left(\frac{\xi}{r^2}\right) = \frac{1}{r^2}\nabla_a\xi - \frac{2}{r^3}r_a\xi$$

and this term is the gauge transformation term for G. The quantity

$$\frac{r^2}{2}\nabla_a G' = \frac{r^2}{2}\nabla_a G - \nabla_a\xi + \frac{2}{r}r_a\xi$$

gives the desired term. Thus I arrive at

$$\overline{K} \equiv K + \frac{1}{2}\ell(\ell+1)G - \frac{2}{r}r^a\left(j_a - \frac{r^2}{2}\nabla_a G\right)$$

Next, I construct a second gauge-invariant quantity from $\overline{p}$. As with $\overline{K}$, the derivative terms are most easily replicated with a covariant derivative of j_a. Then,

$$\begin{aligned}
\overline{p}_{ab} &= p'_{ab} - \nabla_a j'_b - \nabla_b j'_a + ?' \\
&= p_{ab} - \nabla_a \xi_b - \nabla_b \xi_a - \nabla_a \left(j_b - \nabla_b \xi - \xi_b + \frac{2}{r} r_b \xi \right) - \nabla_b \left(j_a - \nabla_a \xi - \xi_a + \frac{2}{r} r_a \xi \right) + ?' \\
&= p_{ab} - \nabla_b j_a - \nabla_a j_b - \nabla_a \left(-\nabla_b \xi + \frac{2}{r} r_b \xi \right) - \nabla_b \left(-\nabla_a \xi + \frac{2}{r} r_a \xi \right) + ?'
\end{aligned}$$

This rather familiar term is, of course, the same one needed for $\overline{K}$. It is convenient to define the quantity

$$\epsilon_a \equiv j_a + \frac{1}{2} r^2 \nabla_a G$$

to simplify these quantities. Then, the gauge-invariant combinations of the scalar fields are

$$\overline{p}_{ab} \equiv p_{ab} - \nabla_a \epsilon_b - \nabla_b \epsilon_a \tag{4.23}$$

$$\overline{K} \equiv K + \frac{1}{2} \ell(\ell+1) G - \frac{1}{r} r^a \epsilon_a \tag{4.24}$$

Finally, consider the case where $\xi = \frac{r^2}{2} G$ Then, $G' = 0$ and

$$\begin{aligned}
j'_a &= j_a - \nabla_a \xi - \xi_a + \frac{2}{r} r_a \xi \\
&= j_a - \nabla_a \left(\frac{r^2}{2} G \right) - \xi_a + \frac{2}{r} r_a \frac{r^2}{2} G \\
&= j_a - \frac{r^2}{2} \nabla_a G - \nabla_a \left(\frac{r^2}{2} \right) G - \xi_a + r r_a G \\
&= j_a - \frac{r^2}{2} \nabla_a G - \nabla_a \left(\frac{r^2}{2} \right) G - \xi_a + r r_a G \\
&= j_a - \frac{r^2}{2} \nabla_a G - \xi_a
\end{aligned}$$

The gauge transformation with components

$$\xi = \frac{r^2}{2} G$$
$$\xi_a = j_a - \frac{r^2}{2} \nabla_a G$$

will always set $j_a = G = 0$, and the gauge-invariant quantities $\overline{p}_{ab} = p_{ab}$ and $\overline{K} = K$ become the only non-zero scalar fields of the tensor spherical harmonic decomposition in the even sector. This is the definition of the RWZ gauge in the even sector, first derived and used by Regge and Wheeler in Schwarzschild coordinates [48].

What remains is to determine the mode-decomposed evolution equations that arise from the Einstein equations. However, this derivation is quite ugly and tedious, and the final generalized forms are very complicated. Surprisingly, however, they can be reduced significantly through the definition of a master function. This Zerilli-Moncrief function is defined as

$$\psi^{\ell m}_{even} \equiv \frac{2r}{\ell(\ell+1)} \left[\overline{K}^{\ell m} + \frac{2}{\Lambda} \left(r^a r^b \overline{h}^{\ell m}_{ab} - r r^a \nabla_a \overline{K}^{\ell m} \right) \right]$$

where

$$\Lambda \equiv (\ell - 1)(\ell + 2) + \frac{6M}{r}$$

where I have reintroduced the ℓ, m notation for the final result [47]. This master function has been published in numerous forms and definitions depending on normalization factor and choice of coordinates, the earliest being Zerilli [49] and Moncrief [50]. This function satisfies the Zerilli equation

$$(\Box - V_{even})\, \psi_{even} = S_{even} \tag{4.25}$$

with potential

$$V_{even} = \frac{1}{\Lambda^2} \left[\mu^2 \left(\frac{\mu + 2}{r^2} + \frac{6M}{r^3} \right) + \frac{36M^2}{r^4} \left(\mu + \frac{2M}{r} \right) \right] \tag{4.26}$$

where $\mu \equiv (\ell-1)(\ell+2)$. Again, I will not derive this equation here, but a detailed derivation of this equation can be found in the paper by Martel [51]. The most surprising feature of the RWZ gauge is that it transforms the gravitational self-force problem into a simple wave equation. All of the scalar fields in the even sector reduce down to requiring the evolution of the wave equation of a single variable ψ.

4.1.2. Odd Parity

The odd parity sector begins very similarly to the even sector, except that the odd sector has far less components. Decomposing the metric perturbation yields

$$\begin{aligned} h_{ab} &= 0 \\ h_{aB} &= \sum_{\ell m} p_a^{\ell m} X_B^{\ell m} \qquad (4.27) \\ h_{AB} &= \sum_{\ell m} p_2^{\ell m} X_{AB}^{\ell m} \qquad (4.28) \end{aligned}$$

The fields $p_a^{\ell m}$ and $p_2^{\ell m}$ are both defined on M^2 and only depend on x^a. As before I examine their behavior under a gauge transformation using Equations 4.7 and 4.8.

$$\begin{aligned} h'_{aB} &= h_{aB} - \nabla_a \Xi_B - D_B \Xi_a + \frac{2}{r} r_a \Xi_B \\ &= \sum_{\ell m} p_a^{\ell m} X_B^{\ell m} - \sum_{\ell m} X_B^{\ell m} \nabla_a \xi^{\ell m} + \frac{2}{r} r_a \sum_{\ell m} X_B^{\ell m} \xi^{\ell m} \\ &= \sum_{\ell m} X_B^{\ell m} \left(p_a^{\ell m} - \nabla_a \xi^{\ell m} + \frac{2}{r} r_a \xi^{\ell m} \right) \end{aligned}$$

$$
\begin{aligned}
h'_{AB} &= h_{AB} - D_A\Xi_B - D_B\Xi_A - 2rr^c\Xi_c\Omega_{AB} \\
&= \sum_{\ell m} p_2^{\ell m} X_{AB}^{\ell m} - 2\sum_{\ell m}\xi^{\ell m} X_{AB}^{\ell m} \\
&= \sum_{\ell m} X_{AB}^{\ell m}\left(p_2^{\ell m} - 2\xi^{\ell m}\right)
\end{aligned}
$$

As with the even sector, these expressions separate by mode, giving the gauge transformations

$$p'_a = p_a - \nabla_a\xi + \frac{2}{r}r_a\xi \tag{4.29}$$

$$p'_2 = p_2 - 2\xi \tag{4.30}$$

To find the gauge-invariant quantity in the odd sector, it is important to recognize that p_2 cannot be the correct starting point because it is impossible to get ξ from p_a. Then, the path to the solution is relatively straightforward.

$$p'_a = p'_a - \nabla_a\left(c_1 p'_2\right) + \frac{2}{r}r_a c_2 p'_2 \tag{4.31}$$

$$p'_a = p_a - \nabla_a\xi + \frac{2}{r}r_a\xi - \nabla_a\left(c_1(p_2 - 2\xi)\right) + \frac{2}{r}r_a c_2(p_2 - 2\xi) \tag{4.32}$$

$$p'_a = p_a - \nabla_a\left(c_1 p_2\right) + \frac{2}{r}r_a c_2 p_2 - \nabla_a\xi + \frac{2}{r}r_a\xi + 2c_1\nabla_a\xi - \frac{4}{r}r_a c_2\xi \tag{4.33}$$

$$p'_a = p_a - \nabla_a\left(c_1 p_2\right) + \frac{2}{r}r_a c_2 p_2 - \nabla_a\xi(1 - 2c_1) + \frac{2}{r}r_a\xi(1 - 2c_2) \tag{4.34}$$

$$\tag{4.35}$$

To get the terms with ξ to cancel, $c_1 = c_2 = \frac{1}{2}$. Then, the gauge-invariant quantity is

$$\overline{p}_a \equiv p_a - \frac{1}{2}\nabla_a p_2 + \frac{1}{r}r_a p_2$$

By choosing $\xi = \frac{p_2}{2}$, p_2 goes to zero and $\overline{p}_a = p_a$. This is the Regge-Wheeler gauge condition in the odd sector.

Finding the evolution equations for the remaining fields requires calculating the mode-decomposed Einstein equations for the metric perturbation in the odd sector. While slightly less painful than the even sector calculations, I again simply refer to the comprehensive paper by Martel and Poisson [47]. In another miraculous turn of events, the odd sector also allows for the definition of a single scalar field, the Cunningham-Price-Moncrief function:

$$\psi^{\ell m}_{odd} \equiv \frac{2r}{(\ell-1)(\ell+2)} \epsilon^{ab} \left(\nabla_a \overline{p}^{\ell m}_b - \frac{2}{r} r_a \overline{p}^{\ell m}_b \right)$$

This master function simplifies the field equations into the Regge-Wheeler equation

$$\left(\Box - V_{odd}\right) \psi_{odd} = S_{odd} \tag{4.36}$$

with potential

$$V_{odd} = \frac{\ell(\ell+1)}{r^2} - \frac{6M}{r^3} \tag{4.37}$$

These equations were first published by Cunningham, Price, and Moncrief [52], and later publications extended these results to frequency domain [53] and eventually to the gauge-invariant formalism of Martel and Poisson [47].

4.2. Numerical Implementation

With the RWZ gauge finally ready, I can now discuss the details of the code itself. The derivation in Section 4.1 reduces the problem of evolving the gravitational fields to that of simulating a scalar wave equation for each ℓ and m mode. The only difference between the scalar self-force and the RWZ self-force is a change in the potential. The scalar wave equation is rearranged to be of the form

$$\partial_t^2 \psi = c_{r_* r_*} \partial_{r_*}^2 \psi + c_{t r_*} \partial_t \partial_{r_*} \psi + c_t \partial_t \psi + c_{r_*} \partial_{r_*} \psi + c\psi + S \tag{4.38}$$

in the tortoise region. The same equation is used in the hyperboloidal regions, except that the derivatives are with respect to the coordinates of the region. While not visible, the c

coefficient contains the potential term. Since Equations 4.25 and 4.36 take on the same general form, they can be treated identically. The only change is in the coefficient c. The non-zero coefficients in the tortoise region are

$$c_{r_*r_*} = 1 \tag{4.39}$$

$$c^{tort}_{even} = -\frac{2(r-2M)}{r^4\left(\lambda r+3M\right)^2}\left(\lambda^2(\lambda+1)r^3+3\lambda^2Mr^2+9\lambda M^2r+9M^3\right) \tag{4.40}$$

$$c^{tort}_{odd} = -\frac{2(r-2M)}{r^4}\left((\lambda+1)r-3M\right) \tag{4.41}$$

where $\lambda = \frac{(\ell-1)(\ell+2)}{2}$. To see the potentials from Equations 4.26 and 4.37, note that $\lambda+1 = \frac{\ell(\ell+1)}{2}$ and that the potential was divided by f^{-1}, the coefficient on the $\partial_t^2\psi$ term. In the hyperboloidal regions, the coefficients become

$$c_{\rho\rho} = \frac{1\mp H}{1\pm H} \tag{4.42}$$

$$c_{\tau\rho} = -\frac{2H}{1\pm H} \tag{4.43}$$

$$c_{\tau} = -\frac{H'}{1\pm H} \tag{4.44}$$

$$c_{\rho} = \mp\frac{H'}{1\pm H} \tag{4.45}$$

$$c^{hyp}_{even} = \frac{1}{1-H^2}c^{tort}_{even} \tag{4.46}$$

$$c^{hyp}_{odd} = \frac{1}{1-H^2}c^{tort}_{even} \tag{4.47}$$

where the upper sign is for the outer region and the lower sign is for the inner region.

Since $H \to \pm 1$ as it approaches $\mathscr{I}^+$ and the horizon, respectively, it is immediately clear that the first four coefficients are finite. Since $H' \to 0$ in both limits, only $c_{\tau\rho}$ and c are non-zero in the limit. The only possible problem is in c, but the quantity $r-2M$ approaches zero faster than $1-H^2$, so all of the coefficients in the principal part are finite. At the

horizon, the only non-zero coefficient is

$$c_{\tau\rho} = 1 \tag{4.48}$$

Approaching $\mathscr{I}^+$, the non-zero coefficients are

$$c_{\tau\rho} = -1 \tag{4.49}$$

$$c = -\frac{\ell(\ell+1)}{2S_+} \tag{4.50}$$

where both the even and odd potentials conveniently have the same limit. To proceed, I implemented these expressions into the self-force code, which is in Fortran. This code uses all of the tricks and techniques discussed in Chapter 3. The effective source code is provided by Barry Wardell and Niels Warburton. I also use initial data for some simulations which is calculated using a frequency domain code written by Seth Hopper. This code is also the source of the frequency domain results with which I compare my code.

4.3. Computational Results

To test and validate the code, I simulate the EMRI system with various choices of DG order and number of elements. I restrict this investigation to circular orbits, as the effective source code currently available can only handle the circular case. The initial data code can already handle eccentric orbits, and my code is ready for validation and testing on both eccentric orbits and the full inspiral. For orbit calculations, a frequency domain code calculates the steady-state solution to the system. In the time domain, this means I force the particle to stay on the orbit and let the simulation converge to a solution. Any initial data in the master functions will propagate out of the domain, and all the quantities of the system will converge to a specific value determined by the behavior of the orbiting particle. The time to convergence can be improved by using initial data calculated by the frequency domain code, but this is only beneficial for the first few ℓ modes. The initial data converges to the final value following a power law with an ℓ-dependent exponent. This means that

the higher modes are fully converged by the time the small ℓ values have converged, even if they use initial data. As such, I use initial data of zero for all higher modes. In an inspiral simulation, the simulation will perform a circular or elliptical orbit until the steady-state solution is reached. Then, the particle is allowed to begin the inspiral. This method works because the early inspiral is well-approximated by the adiabatic approximation, which is effectively what this method generates as initial data for the inspiral.

My simulations evolve the EMRI and output the energy and angular momentum fluxes through the horizon and out to $\mathscr{I}^+$. While I have implemented output for reconstruction of the metric perturbation, I do not currently have data with which to compare. The full self-force also cannot be constructed because I only have $\ell \geq 2$ in the simulation. As mentioned in Section 4.1, the master functions are only valid for these modes. The $\ell = 0$ and $\ell = 1$ modes must be handled separately, and this is not yet implemented. For all simulations, the point mass is confined to a circular orbit with $r_0 = 10M$ around a Schwarzschild black hole. As mentioned before, the initial data reduces the runtime, so I use initial data for $\ell = 2$ to $\ell = 5$ and include modes up to $\ell = 40$. This simulation is evaluated up to $T = 1000M$. The initial data is meant only to speed up the simulation, so I also simulate the first five modes with no initial data for $T = 10,000M$ to verify that the same result is reached.

In addition to fluxes, calculation of the polarizations of the gravitational waves are also important. However, I do not examine these quantities here. While I have implemented the calculations for the gravitational wave polarizations, I do not currently have comparison data, so I cannot use them for validation. The expressions for the polarizations and fluxes at $\mathscr{I}^+$ are found by expanding the perturbation in powers of r^{-1} and examining the behavior of ψ at the boundaries [47]. After deriving the piece of the perturbation which represents the radiative field, the two polarizations of the gravitational waves and the fluxes can be derived [54]. At the horizon, only the fluxes need to be calculated, and those expressions are derived by evaluating the field equations at $r = 2M$ and integrating over the sphere [55]. Conveniently, the expressions for the fluxes are identical for both boundaries.

The polarizations are given by

$$h_+ = \frac{1}{r}\sum_{\ell m}\left[\psi^{\ell m}_{\text{even}}\left(\partial_\theta^2 + \frac{\ell(\ell+1)}{2}\right)Y^{\ell m} - \frac{im}{\sin(\theta)}\psi^{\ell m}_{\text{odd}}\left(\partial_\theta - \cot(\theta)\right)Y^{\ell m}\right] \tag{4.51}$$

$$h_\times = \frac{1}{r}\sum_{\ell m}\left[\frac{im}{\sin(\theta)}\psi^{\ell m}_{\text{even}}\left(\partial_\theta - \cot(\theta)\right)Y^{\ell m} + \psi^{\ell m}_{\text{odd}}\left(\partial_\theta^2 + \frac{\ell(\ell+1)}{2}\right)Y^{\ell m}\right] \tag{4.52}$$

and the fluxes are given by

$$\left\langle \frac{dE}{dv} \right\rangle = \frac{1}{64\pi}\sum_{\ell m}(\ell-1)(\ell+1)(\ell+2)\left\langle |\dot{\psi}^{\ell m}_{\text{even}}|^2 + |\dot{\psi}^{\ell m}_{\text{odd}}|^2 \right\rangle \tag{4.53}$$

$$\left\langle \frac{dJ}{dv} \right\rangle = \frac{1}{64\pi}\sum_{\ell m}(\ell-1)(\ell+1)(\ell+2)\left\langle \bar{\psi}^{\ell m}_{\text{even}}\dot{\psi}^{\ell m}_{\text{even}} + \bar{\psi}^{\ell m}_{\text{odd}}\dot{\psi}^{\ell m}_{\text{odd}} \right\rangle \tag{4.54}$$

Here, the coordinate v is from the coordinates (v, r, θ, ϕ). At $\mathscr{I}^+$, this represents the outgoing Eddington–Finkelstein coordinates ($v = t + r_*$) and at the horizon this represents the ingoing Eddington–Finkelstein coordinates ($v = t - r_*$). Traditionally, the ingoing coordinates uses u instead of v, but I do not give Equations 4.53 and 4.54 twice since they have an identical form for both boundaries. The fact that these expressions are in a different coordinate system may initially seem problematic, but the time coordinate of the hyperboloidal coordinate τ asymptotes to the Eddington–Finkelstein coordinates u and v, respectively. At the boundaries the time derivatives in these coordinates are identical and may be evaluated without worry. Also, these equations appear to mix even and odd ψ at first glance, but the even (odd) field is zero when the mode is odd (even). Therefore, any given (ℓ, m) mode will only have a single term.

To check convergence, I vary the number of elements and DG order. The number of elements used are 16, 32, and 64. The DG orders used are 4, 6, 8, 10, 12, and 16. The number of elements in the tortoise region—the *t_size* parameter in the code—varies with the number of elements. I use *t_size* values of 5, 10, and 20, respectively. I set $S_- = 20$, and the location of the particle in tortoise coordinates is $R_* = 12.7725887222398$. With these parameters,

this sets $S_+ = 45.545177444479563$. The location of the boundary between the hyperboloidal layers and the tortoise region are $T_- = -7.7102792291600828$ and $T_+ = 33.255456673639642$.

Once the simulation has run for long enough to reach a stable solution, I average over the last 50 data points of output. I take these results and compare with the frequency domain calculations by plotting the absolute difference between them. Figures 4.1, 4.2, and 4.3 show the absolute difference between the energy flux calculated by my code and the frequency domain results as computed by Seth Hopper's code. Figures 4.4, 4.5, and 4.6 show the absolute difference between the angular momentum flux calculated by my code and the frequency domain results as computed by Seth Hopper's code. The plots are divided by the number of elements, and all show multiple lines for the different DG orders. The data points marked with a '$\circ$' used initial data, and data points marked with a '+' did not.

The reason I do not use the relative difference is because many high-order modes are effectively zero for the circular case. Due to the numerical methods, my code will produce a value on the order of 10^{-20} to 10^{-30} depending on the resolution of the DG parameters, while the frequency domain code will generally get values from 10^{-34} all the way down to 10^{-90}. While both of these are effectively zero, it makes producing a meaningful relative difference plot difficult.

In these plots, the time domain results clearly converge to the frequency domain results as the DG order and element count are increased. The 16 element simulations display this most clearly, as that is a very small number of elements. Since the number of elements is low, increasing the DG significantly improves the accuracy of the code. As the number of elements increases, increasing the order has less and less effect. In Figure 4.3, the high ℓ-mode data at the horizon has completely converged, and even the 4th-order simulation is nearly identical to the higher order simulations.

To reproduce these results one parameter is of particular importance. For modes without initial data, the effective source is turned on smoothly so that there isn't a sudden discontinuity introduced by using zero initial data. In practice, not doing so with the scalar self-force

resulted in the excitation of unphysical modes, so it is also used here. The speed at which this is turned on is controlled by the parameter $tsigma$. I use $tsigma = 10^{-6}$ to rapidly turn on the source. I first performed these calculations with a larger $tsigma$. However, this results in higher error in the initial time. A similar numerical error would arise from turning the effective source on smoothly with initial data. Since the initial data from the frequency domain code is close to the correct solution for the full source, turning it on smoothly effectively uses the wrong source temporarily, causing the simulation to take longer to settle down to the correct solution and defeating the purpose of using initial data.

The final steady state solution for the energy flux is unaffected by numerical error from turning on the source, but the angular momentum flux accumulates error from the transitional phase. The reason for this is that the source of error is only present in the angular momentum flux equation. In Equation 4.54, the expressions include ψ itself, not just the time derivative. This quantity is obviously just an integral of $\dot{\psi}$ over the simulation (plus any initial value). However, $\dot{\psi}$ only settles down to the correct value once any bad initial data propagates out of the domain. During this time, the incorrect $\dot{\psi}$ accumulates error in ψ, which then appears in the angular momentum flux. This error would also be evident in the gravitational wave polarizations, as they also depend on ψ. This is not visible in Figures 4.4-4.6 at $\mathscr{I}^+$ because the error which is introduced is significantly smaller than the final values for the angular momentum flux. Thus, using initial data to 'prime' the simulation not only improves runtime but also reduces a source of error in the code. To minimize this effect on the modes using zero initial data, the value of tsigma should be set to be small so that the initial transient source quickly becomes the full source while still remaining smooth. Since the initial values are zero or near-zero, the errors introduced during the transient source phase are minimal so long as $tsigma$ is sufficiently small. For these simulations, I set $tsigma = 10^{-6}$ for all of these simulations.

There are also several unusual features I should explain in these plots. First, The horizon data plot in Figures 4.3 and 4.6 show the lowest order method appearing to achieve the

highest accuracy. This is actually because the high ℓ modes are close to zero, and the difference between the two solutions is limited by how small the time domain value can get during the run. In the case of the lowest resolution, the numerical inaccuracy actually leads to the flux dropping to zero faster, which is closer to the frequency domain results than the higher order simulations. Second, the jagged nature of some of the data points is because the data from my code and the frequency domain code matched for all the outputted digits. When that happens, I approximate the difference as being one digit beyond the available data. Finally, the simulations using initial data are supposed to get the same result as with no initial data but take less time to run. However, the higher order simulations show the data using initial data to level off in accuracy. This is especially noticeable in the horizon data plot in Figures 4.2 and 4.3. This happens because the initial data code also has numerical error. At some point, the inaccuracy of the initial data becomes the dominant error in the final results, causing the final result to differ from the results without initial data. The 'bad' initial data has to propagate completely out of the domain, which can take longer than just starting with zero initial data. If the initial data simulation were run for as long as the simulation with no initial data these data points should converge, but the proper solution is to calculate initial data to an accuracy appropriate for the simulation.

These results show that my time domain code agrees with highly accurate and well-tested frequency domain codes for circular orbits and is ready for eccentric orbits. The only changes to go from circular orbits to eccentric orbits and inspirals is the effective source code. Testing for eccentric orbits proceeds similarly to the circular orbit. Other than a new effective source, the only remaining task before testing full inspirals simulations are possible is the implementation of evolution equations for the $\ell = 0$ and $\ell = 1$ modes. Once these modes are implemented, the full self-force calculation can be compared for circular and eccentric orbits as a final validation before inspiral simulations begin.

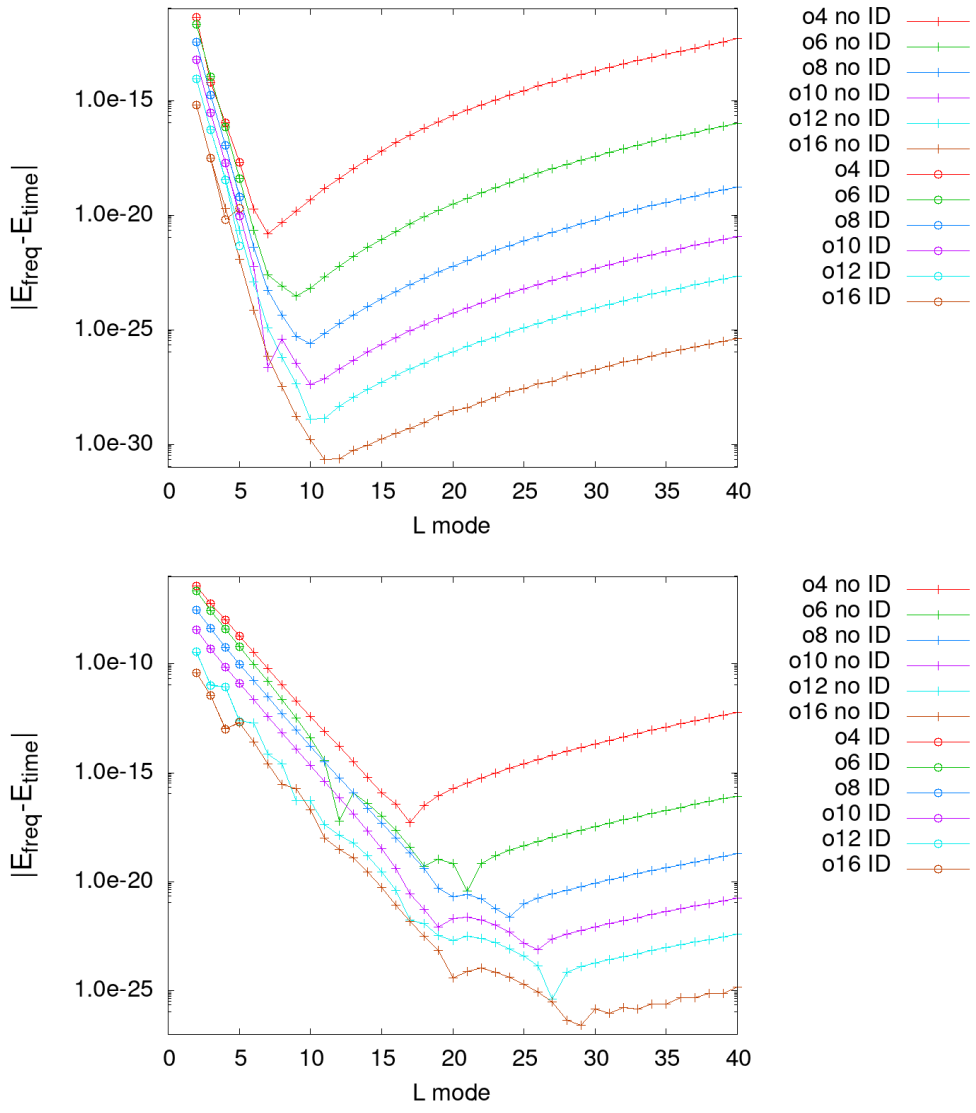

Figure 4.1. The absolute difference between the calculated energy flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 16 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

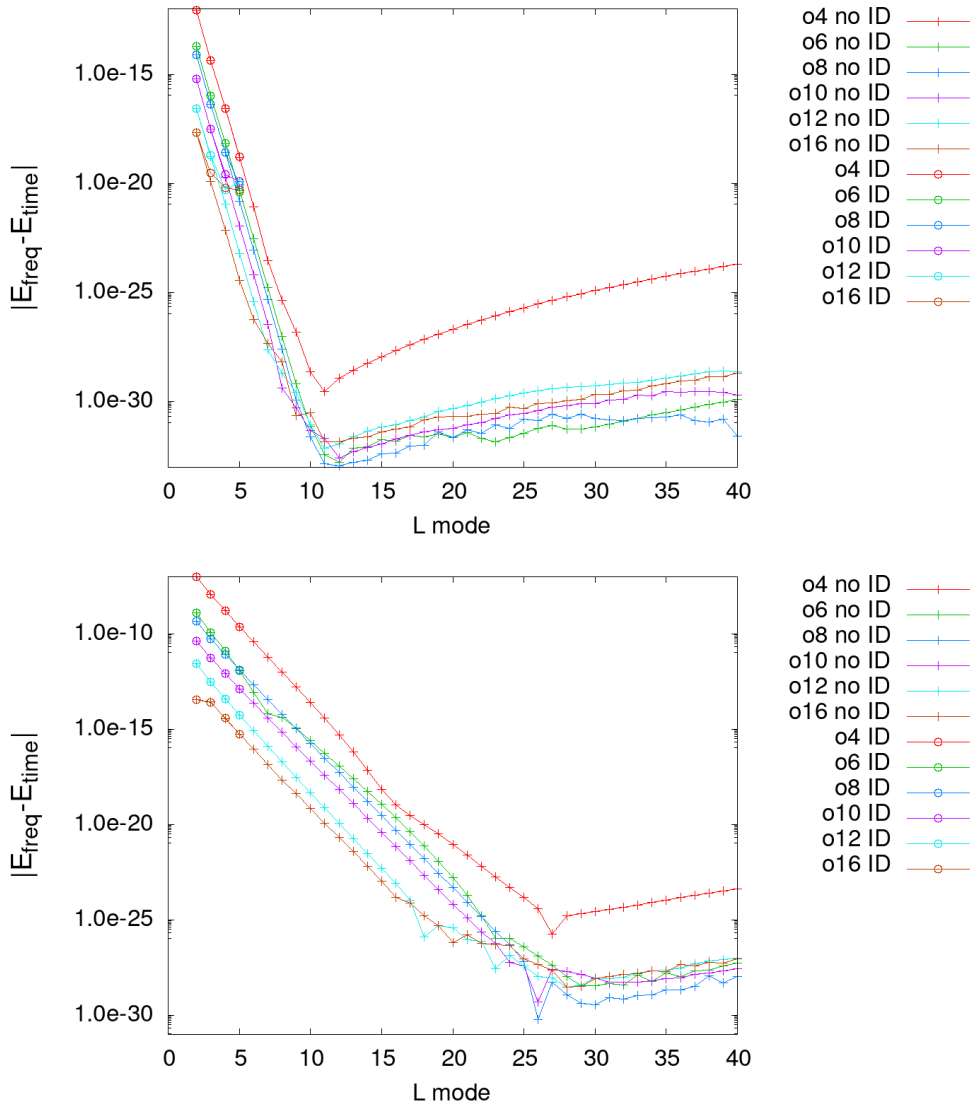

Figure 4.2. The absolute difference between the calculated energy flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 32 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

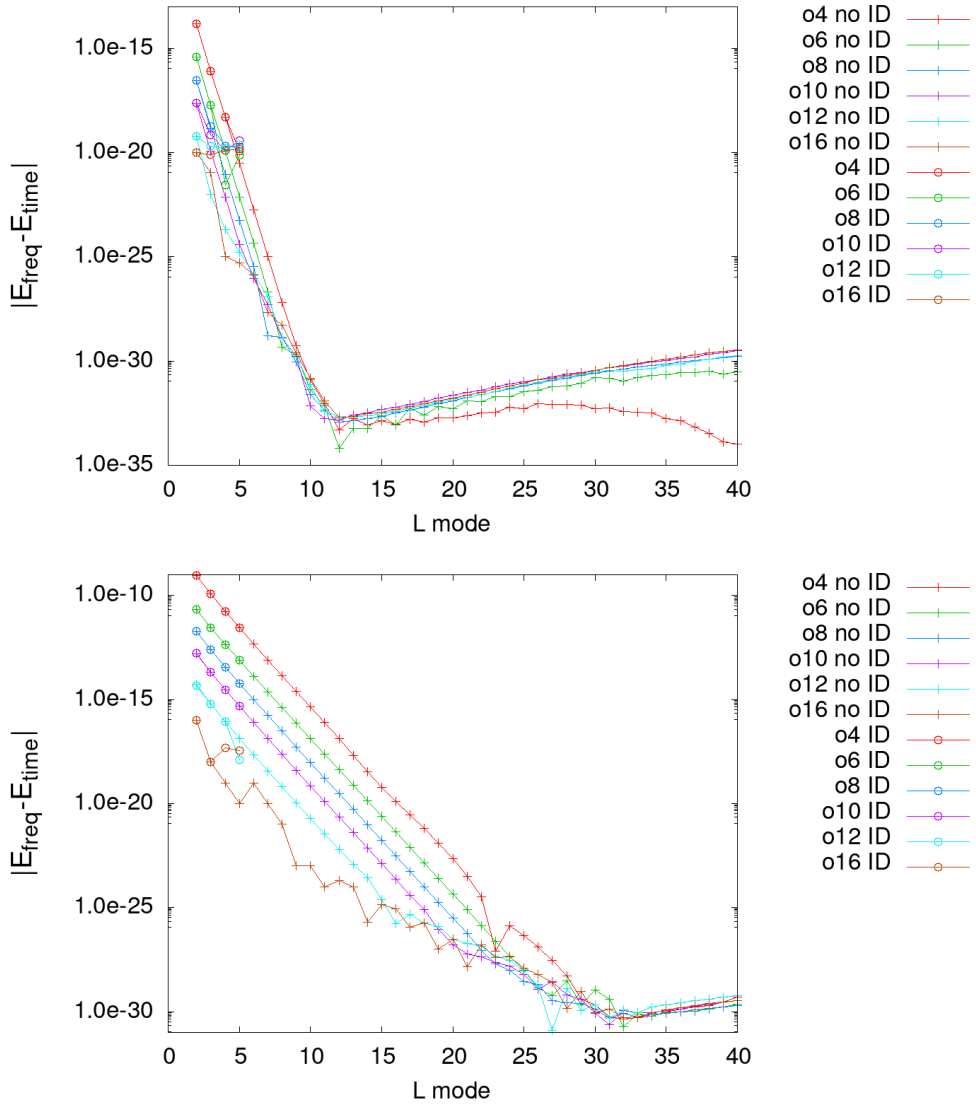

Figure 4.3. The absolute difference between the calculated energy flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 64 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

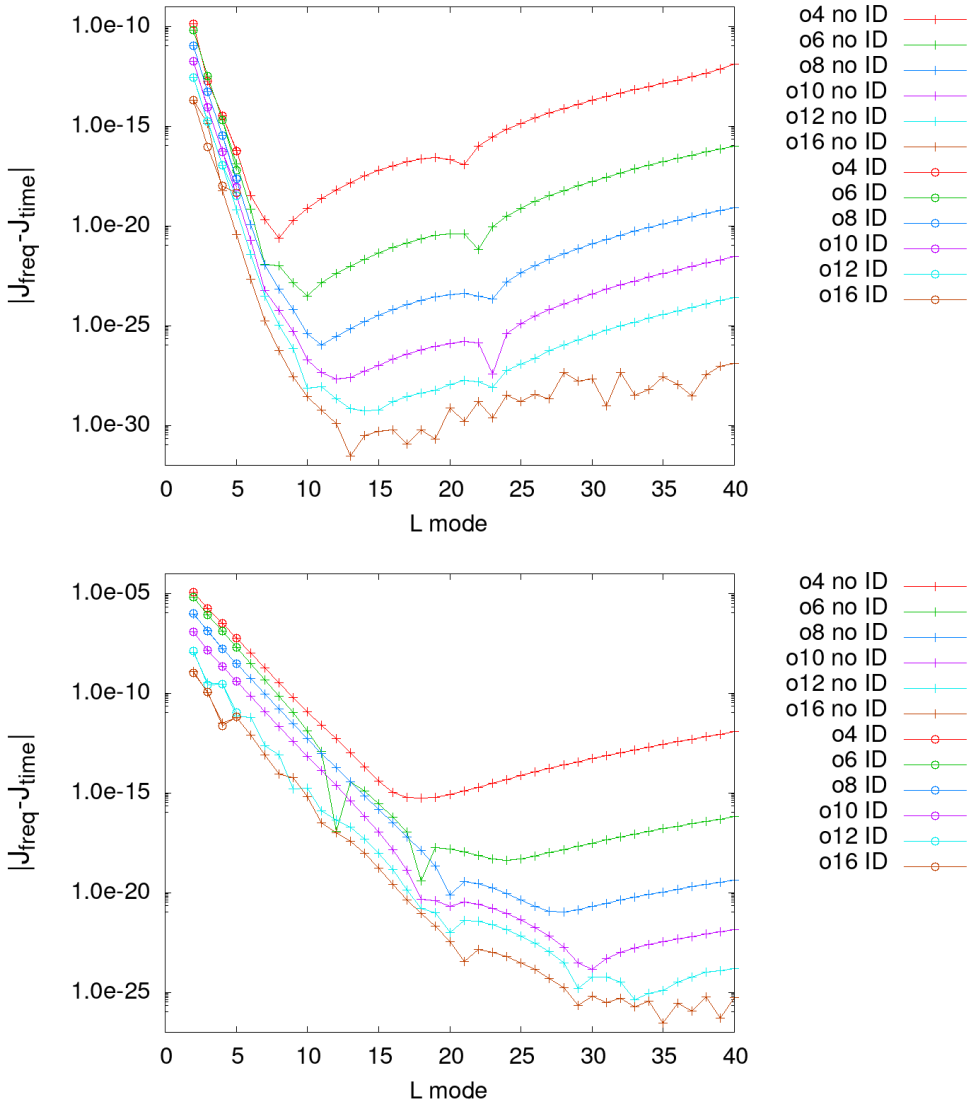

Figure 4.4. The absolute difference between the calculated angular momentum flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 16 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

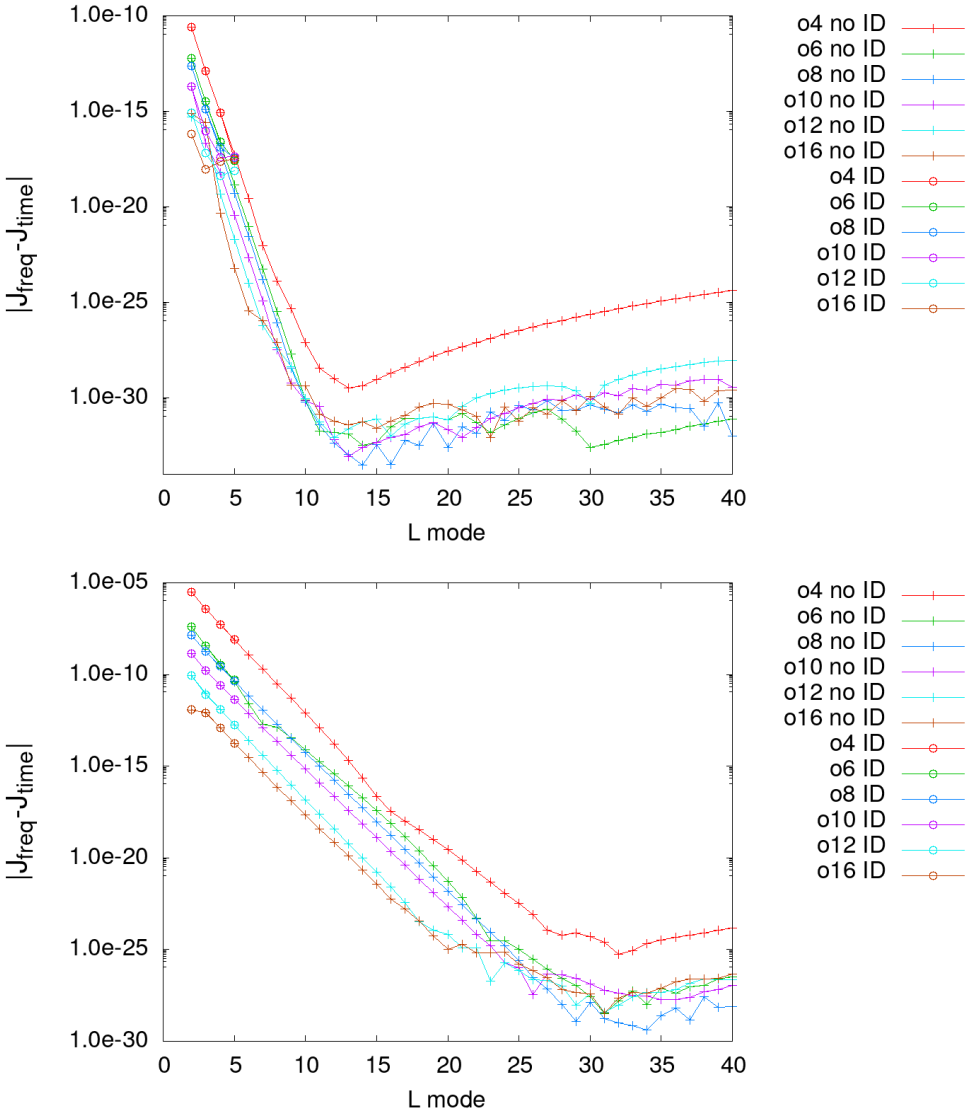

Figure 4.5. The absolute difference between the calculated angular momentum flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 32 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

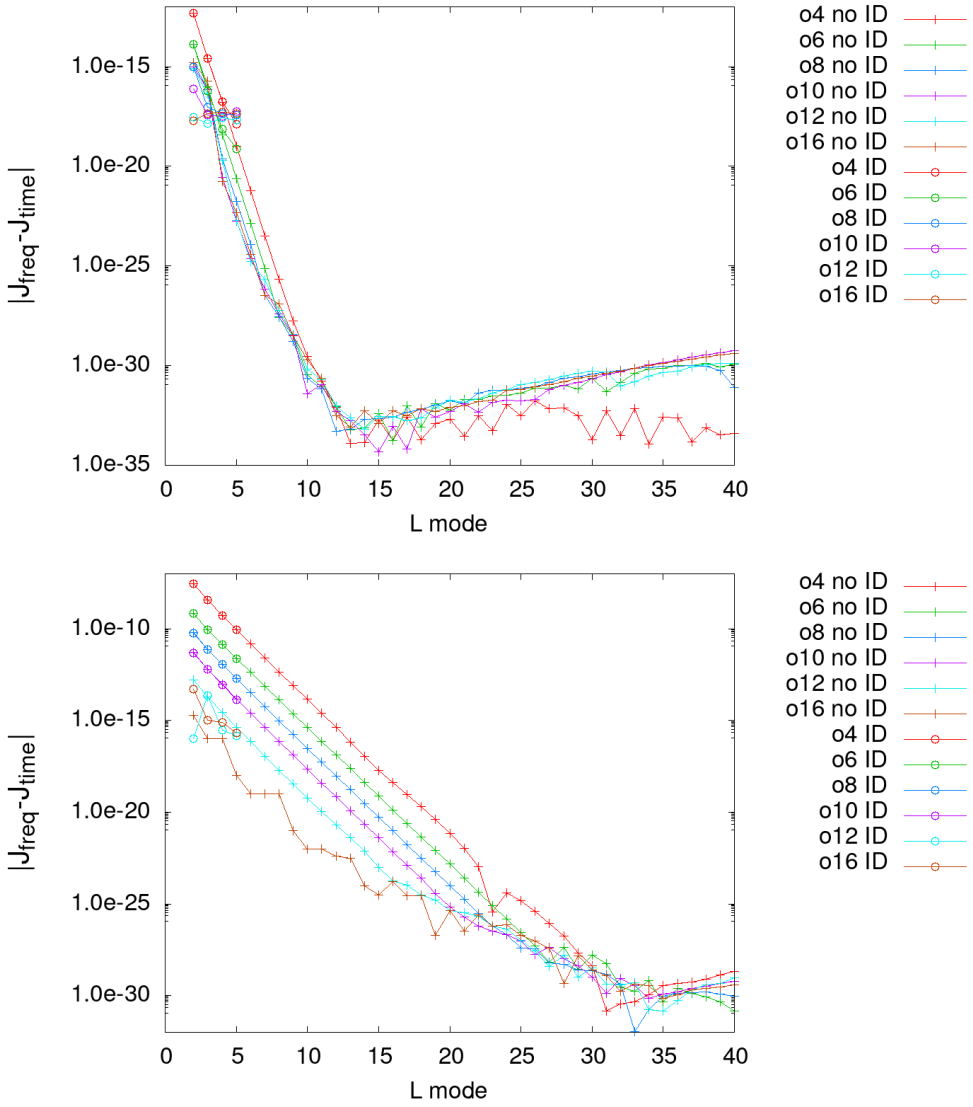

Figure 4.6. The absolute difference between the calculated angular momentum flux and the frequency domain results at the horizon (top) and $\mathscr{I}^+$ (bottom) as function of ℓ with 64 DG elements at different DG order (purple: 8, green: 10, blue: 12, orange: 14 and yellow: 16). The points marked with a '$\circ$' used initial data, and those marked with a '+' did not use initial data.

Chapter 5. Self-force Implementation in the Lorenz Gauge

As discussed in Chapter 4, I have developed a code to simulate the gravitational self-force in the Regge-Wheeler-Zerilli gauge. This naturally leads to a preliminary question. Since the RWZ equations are developed and implemented, why bother developing evolution equations in an entirely new gauge? There are several reasons for this. First, the implementation of multiple gauges provides further validation of the numerical methods used. Particularly, any observable quantities are gauge-invariant, so having results in multiple gauges can provide a more rigorous test of validity. These comparisons are difficult in practice, but they can be done with sufficient effort. Secondly, work on second-order self-force by the EMRI community is furthest along in the Lorenz gauge. As mentioned in Section 3.4, LISA requires second-order contributions, so I must also consider what will provide the smoothest transition to second-order simulations in the future. While the RWZ gauge is quite convenient at first-order, that convenience has yet to manifest at second-order. Thus, the Lorenz gauge appears to be a more 'natural' choice, so it should be developed in preparation for the possibility that it will be needed at second-order.

5.1. Tensor Spherical Harmonics

The derivation for the Lorenz gauge begins similarly to the RWZ gauge—with a tensor spherical harmonic decomposition. The primary difference is that where the RWZ gauge simplifies the evolution equations by choice of gauge, the Lorenz gauge does not. Hence, the evolution equations are far less elegant. However, the general methods are quite similar. The Lorenz gauge is defined as $\nabla^{\beta}\tilde{g}_{\alpha\beta} = 0$, and it is more convenient to consider the trace-reversed metric perturbation

$$\bar{h}_{\alpha\beta} = h_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} \tag{5.1}$$

Then, the metric perturbation equations reduce to

$$\Box \bar{h}_{\alpha\beta} + R^{\mu}{}_{\alpha}{}^{\nu}{}_{\beta} \bar{h}_{\mu\nu} = -16\pi T_{\alpha\beta}. \tag{5.2}$$

I then decompose this equation using tensor spherical harmonics. I am not using the same formalism as in Chapter 4, as this section builds off of the work of Barack, Lousto, and Sago [56, 57] as its basis. In the case of the RWZ gauge, the details of the form of the tensor spherical harmonics do not matter because all ten of the scalar fields generated by the decomposition reduce to a single scalar field ψ. In contrast, the Lorenz gauge will not remove all of the different scalar fields in the decomposition, and I will instead find ten coupled evolution equations for each (ℓ, m) mode. The exact form of the equations will depend on choice of decomposition, so I must be more explicit in my choice of basis. I start with the basis used by Barack and Sago [57], which I will refer to as the BLS basis. In Schwarzschild coordinates, the trace-reversed metric perturbation is related to the tensor harmonics by the equation

$$\bar{h}_{\alpha\beta} = \frac{m}{r} \sum_{\ell m} \sum_{i=1}^{10} a^{(i)\ell} h^{(i)}_{BS}(t,r) Y^{(i)\ell m}_{\alpha\beta}(\theta,\phi;r) \tag{5.3}$$

where

$$a^{(i)\ell} = \frac{1}{\sqrt{2}} \begin{cases} 1, & i \in \{1,2,3,6\} \\ \left[\ell(\ell+1\right]^{-1/2}, & i \in \{4,5,8,9\} \\ \left[\ell(\ell-1^2)(\ell+2)\right]^{-1/2}, & i \in \{7,10\} \end{cases}$$

The first seven $h^{(i)}$ correspond to the even-sector fields in Section 4.1.1, and the last three correspond to the odd-sector fields in Section 4.1.2. The tensor spherical harmonics $Y^{(i)\ell m}_{\alpha\beta}$ here are the same as those used by Barack and Sago [57], and they are included in Appendix A. This differs from the Barack and Lousto basis [56] by applying the rescaling of

$$\bar{h}^{(3)}_{BS} \equiv f^{-1} \bar{h}^{(3)}_{BL} \tag{5.4}$$

This rescaling was done because the $\bar{h}^{(3)}_{BL}$ field vanishes at the horizon. I verified during my experimentation that this change resolves several singular components of the coupling matrix

that are present if the original basis is used. It is also at this point where I diverge from the work of Barack, Lousto, and Sago. I will discuss the point at which their basis breaks later, but I found that it is impossible to generate well-behaved evolution equations in the inner hyperboloidal region using the BLS basis because some of the coupling coefficients will inevitably be singular at the horizon. No amount of fiddling with constraints could properly resolve this issue, leading to my adoption of a new basis. For the remainder of the paper, I use the following basis:

$$h^{(1)} \equiv h^{(1)}_{BLS} + h^{(2)}_{BLS} \tag{5.5}$$

$$h^{(2)} \equiv h^{(1)}_{BLS} - h^{(2)}_{BLS} \tag{5.6}$$

$$h^{(4)} \equiv h^{(4)}_{BLS} + h^{(5)}_{BLS} \tag{5.7}$$

$$h^{(5)} \equiv h^{(4)}_{BLS} - h^{(5)}_{BLS} \tag{5.8}$$

$$h^{(8)} \equiv h^{(8)}_{BLS} + h^{(9)}_{BLS} \tag{5.9}$$

$$h^{(9)} \equiv h^{(8)}_{BLS} - h^{(9)}_{BLS} \tag{5.10}$$

The scalar fields $h^{(3)}$, $h^{(6)}$, $h^{(7)}$, and $h^{(10)}$ are unchanged from the BLS basis.

5.2. Evolution Equations

After choosing a basis, I insert Equation 5.3 into Equation 5.2 to generate evolution equations for each $h^{(i)}$. This requires finding linear combinations of the equations such that only a single $\partial_t^2 h^{(i)}$ appears in each equation (or $\partial_\tau^2 h^{(i)}$ in hyperboloidal coordinates). This term remains on the left-hand side while all other terms become the right-hand side for the numerical implementation. Finding the correct combination for each evolution equation is rather straightforward, and these combinations are given in Appendix B. In the tortoise

region, these equations take the form

$$\partial_t^2 h^{(i)} = c_{r_* r_*} \partial_{r_*}^2 h^{(i)} + c_{t r_*} \partial_t \partial_{r_*} h^{(i)} + c_t \partial_t h^{(i)} + c_{r_*} \partial_{r_*} h^{(i)} + c h^{(i)} + \sum_{i=1}^{10} M_{(j)}^{(i)} h^{(j)} + S \tag{5.11}$$

where S is the source term and $M_{(j)}^{(i)}$ is the coupling matrix. This matrix is an operator and contains derivatives, as the fields also couple to the first derivatives of the other fields.

The various c coefficients come from the principal part ($\Box \bar{h}_{\alpha\beta}$) of the equation and do not depend on i. The potential term is contained within the c coefficient. Since the principal part is unchanged from the RWZ gauge, the first four coefficients are identical and are given by Equation 4.39 in the tortoise region and Equations 4.42–4.45 in the hyperboloidal layers. The remaining potential term is

$$c_{tort} = -\frac{r - 2M}{r^3} \left(\ell(\ell+1) + \frac{2M}{r} \right)$$

in the tortoise region and

$$c_{hyp} = -\frac{1}{1 - H^2} c_{tort}$$

in the hyperboloidal regions. In the limits, the coefficients are given by Equation 4.48 at the horizon and Equations 4.49 and 4.50 at $\mathscr{I}^+$. While the potentials are different, their values in the limits are once again identical.

The final step is to find the limits for the coupling matrix. This step is where the BLS basis results in singular coefficients at the horizon. In the limit approaching $\mathscr{I}^+$, the coupling matrix is finite. The limit towards the horizon is unfortunately not. The division by $1 - H^2$ (which is zero in the limit) creates this issue, as some terms do not have a corresponding term in the numerator to counteract it. In the BLS basis, the singularities appear in equations for $i \in \{1, 2, 4, 5, 8, 9\}$. This could potentially be solved by using the constraint equations

to cancel out these terms. However, an additional factor interferes with this method for resolving the singular terms.

5.3. Constraint Damping

When selecting a gauge, two options are available. One is to use the degrees of freedom to eliminate variables. This is what the RWZ gauge does, reducing the total number of scalar fields from ten to two. The other option is to keep all the variables along with the constraint equations. This implementation of the Lorenz gauge keeps all ten scalar fields along with four constraint equations. Ideally, these should all be satisfied throughout the simulation to ensure the validity of the results. Part of this is finding a way to calculate or approximate constraint-satisfying initial data. However, even with good initial data numerical error can introduce constraint violations during a simulation.

To ensure that the constraints remain satisfied, I add constraint damping terms to the evolution equations so that any constraint violations which occur will rapidly return to zero. As a reminder, the four constraint equations are derived from the gauge condition $Z_\alpha \equiv \nabla^\beta \bar{h}_{\alpha\beta} = 0$. To introduce constraint damping, Barack and Lousto[56] propose adding term to Equation 5.2

$$-\kappa \left(t_\alpha Z_\beta + t_\beta Z_\alpha \right) \tag{5.12}$$

where κ is a positive constant and t_α is a future-directed timelike vector field, which is shown to damp the constraints by Gundlach *et al* [58]. Adding this damping term (which is equal to zero in the continuum limit) drives any changes of the constraints from zero back towards zero, helping to stabilize the code.

This addition also has no effect on the principal part of the equation, so the only part that changes is the coupling matrix $M^{(i)}_{(j)}$. This also means that the construction devised for

the evolution equations is also still valid. Barack and Lousto choose for their damping

$$\kappa = -f' \tag{5.13}$$

$$t_\alpha = \left(1, f^{-1}, 0, 0\right) \tag{5.14}$$

$$\tilde{Z}_\alpha = (f Z_r, Z_r, Z_\theta, Z_\phi) \tag{5.15}$$

$$\tag{5.16}$$

which violates the requirement that κ be constant. They validate this choice by providing numerical evidence for the damping effect and argue that so long as κ varies slowly with respect to the background curvature the damping effect should not be affected [56]. It also violates the requirement that κ be positive, but that is not discussed. They also choose $t_\alpha = (1, f^{-1}, 0, 0)$, which is lightlike instead of timelike. This might interact with κ such that the sign change is not a problem, but that is not analytically proven. Finally, Z_α is replaced with $\tilde{Z}_\alpha$. These choices are all likewise defended on the basis of the numerical evidence of damping. The paper differs from the values I have here because of several typos. I have verified the quantities reported here after discussions with Barack and rederiving the coupling coefficients in the Schwarzschild coordinates using the original BLS basis, with one exception. I found that to get the same equations they report I must set $\kappa = 0$ for the $i = 3$ equation. Since this term is zero in the continuum limit, there is no problem with doing so.

I use the same t_α as in Equation 5.14 and transform it into the new coordinates. My κ differs from Equation 5.13, and I also choose a different κ for $i = 3$, hereafter referred to as κ_3. The reason for this is that the term that arises in this equation requires an f in the inner hyperboloidal region to remain finite at the horizon. This did not appear in the BLS equations because $\kappa_3^{BLS} = 0$. To preserve the proper units, I use f/r so that it matches the units of f'. Instead of $\tilde{Z}_\alpha$, I just use Z_α. I give the values for t_α, κ, and κ_3 in Table 5.1. It is also important to note that Barack, Lousto, and Sago work in Schwarzschild coordinates,

Table 5.1. Values used for damping in the various coordinate regions. The κ values in the inner layer are complicated and given in Table 5.2.

	Inner Layer	Tortoise Region	Outer Layer
t_α	$\left(1, \frac{1+H}{1-H}, 0, 0\right)$	$(1, 1, 0, 0)$	$\left(1, \frac{1+H}{1-H}, 0, 0\right)$
κ	Varies	$-f'$	$-f'$
κ_3	Varies	$-f/r$	$-(1-H)f/r$

while I am working in tortoise coordinates. This leads to the apparent difference in our definitions of t_α.

The behavior of damping in the inner layer is quite finicky, and I will discuss this in detail, but first I must finally deal with the issue of singular terms in the coupling matrix. The constraint damping may not seem to relate to resolving the singular terms, but I can use these equations to cancel out the singular terms. Since the constraints are zero, I am free to add them to the equations as needed. Then, I could potentially find a combination which will cancel out the bad terms exactly. However, a subtlety arose when attempting to cancel these out in the BLS basis. I can, in fact, cancel out the terms in the BLS equations using the constraint equations. However, upon implementing them and running the simulation, I saw that the constraints were rapidly violated, and that those violations were not damped at all. Further analysis shows that the terms I need to add to the evolution equations for $i \in \{1, 5, 9\}$ are the same terms which arise from the constraint damping, but with the sign switched. Worse, they die off slower than the damping terms, meaning that near the horizon constraint violations were actually *amplified* instead of damped! This clearly causes numerical instability is not any better than having singularities.

These difficulties led to my search for a new basis, which culminated in my choice of the basis given in Equations 5.5–5.10. The singular terms in the new evolution equations can be

canceled by adding the following to the respective right-hand side:

$$\begin{aligned} eq^{(2)} &= eq^{(2)} - \frac{2H}{1-H^2} Z_\tau \\ eq^{(5)} &= eq^{(5)} - \frac{2H}{1-H^2} Z_\theta \\ eq^{(9)} &= eq^{(9)} - \frac{2H}{1-H^2} Z_\phi \end{aligned} \tag{5.17}$$

This results in finite values at the horizon without ruining the constraint damping. As for the damping itself, simply choosing $-f'$ proved insufficient. Some of the equations required a factor of $(1 + H)$ in κ to keep the term finite at the horizon. In addition, the terms rapidly approach zero as they approach the horizon. In my testing, I found that numerical instabilities would arise very near the horizon and grow uncontrollably. To remedy this, I tested with the κ values given in Table 5.2. I introduce the positive constants c_1, c_2, and c_3

Table 5.2. Expressions used for κ in the inner layer. The constants c_1, c_2, and c_3 are variables in the code that can be changed for experimenting with damping.

	$i = 1$	$i = 3$	$i \in \{4, 8\}$
κ	$-f'(1 - c_1 H)$	$-f(1 + H)(1 - c_2 H)$	$-f'(1 + H)(1 - c_3 H)$

to allow for testing in the code with different values. Since $H \to 0$ at the the layer boundary, these are all continuous with the damping in the tortoise region. At the horizon, $H \to -1$, and the c_i factors are all just positive multiplicative factors.

After canceling the singular terms and introducing constraint damping, the evolution equations are finally ready to be implemented into the code. For completeness, I provide the full form of the coupling matrix $M^{(i)}_{(j)} h^{(j)}$ for the tortoise and inner hyperboloidal layers in Appendix C. The outer hyperboloidal layer immediately follows from the tortoise matrix, and this relation is also described in the Appendix.

5.4. Methodology and Results

To derive these equations, I use Mathematica to apply the tensor spherical harmonic decomposition, solve for the evolution equations, and apply the limits. The notebook then

automatically generates Fortran code which I incorporated into the same code framework used for the RWZ code. This code is noticeably more complicated than the RWZ code, however, as it contains ten coupled equations for each ℓ, m mode instead of just one. This is part of the motivation for using Mathematica to generate code pieces, as it removes the human error of manually typing many long, complicated expressions.

The long-term goal of this code is to be able to simulate the gravitational self-force for an inspiraling object. However, the first test of code validity is to examine how initial data will evolve without a source. Without a source, the initial data simply propagates into the horizon and out to $\mathscr{I}^+$, eventually decaying to zero. This decay is equivalent to the behavior of the black hole post-merger where only the central black hole remains. Since a single black hole can only have mass and spin, any ℓ modes higher than $\ell = 1$ must decay away via gravitational waves. This part of the gravitational wave signal is referred to as the quasinormal mode (QNM) ringdown. The QNM ringdown has a specific behavior that known to very high accuracy, and this behavior is determined purely by the background spacetime. For the Schwarzschild metric, each ℓ mode oscillates with a specific complex frequency ω. Berti, Cardoso, and Starinets provide a detailed overview of quasinormal modes in all its flavors, along with an extensive literature review [59]. For my purposes, I am using the QuasiNormalModes Mathematica package from the Black Hole Perturbation Toolkit to generate these values [60].

I assess my code's current performance via several metrics. First, I validate that it is stable for long after the initial data has propagated out of the domain. This is to ensure that the constraint damping is working properly. Second, I examine the wave behavior as it propagates out, especially across the boundaries between the coordinate regions. Third, I check that the constraints remain approximately zero for the entire runtime (in the case of constraint-satisfying data) or decays rapidly (for simple gaussian initial data). Finally, I approximate the QNM frequencies of my data and compare them to frequency domain calculations.

5.4.1. Numerical Stability

I perform all my analysis with both constraint-satisfying initial data and constraint-violating initial data. As the code's behavior is generally the same in both, I show the data for the constraint-satisfying initial data unless specifically mentioned. I solve for the constraint-satisfying initial data using the constraints and setting

$$h^{(i)} = 0,\ i \notin \{4, 6, 10\} \tag{5.18}$$

$$\partial_t h^{(i)} = \Delta,\ i \notin \{4, 6, 10\} \tag{5.19}$$

where

$$\Delta = AExp\left[-\frac{(r_* - r_0)^2}{2\sigma^2}\right]$$

I then solve for the remaining scalar fields by enforcing $Z_\alpha = \partial_t Z_\alpha = 0$. The derived initial data expressions are

$$\begin{aligned} h^{(4)} &= \frac{2M - 3r}{f}\Delta \\ \partial_t h^{(4)} &= -\left(1 + \frac{2M}{r} + \frac{4M}{r - 2M}\right)\Delta \\ h^{(10)} &= -\frac{2r}{f}\Delta \\ \partial_t h^{(10)} &= 0 \end{aligned}$$

The expressions I derive for $h^{(7)}$ and its derivative via this method are very complicated, so I do not reproduce them here. The constraint-violating initial data uses the initial data in Equations 5.18 and 5.19 for all modes.

Evolving with only the damping used in the other regions results in a very unstable simulation. The scalar fields rapidly become unstable in the inner hyperboloidal region. As an attempt to counteract this, I introduced the c_i given in Table 5.2. The constraint damping is meant to help with stability, but many of those terms go to zero in the inner

layer. The c_i introduce multiplicative factors with the goal of intensifying the damping in the layer and improve the stability. I tested different combinations of these quantities to try to find a combination which resulted in stable evolution. Even if this is the correct solution, the choice of c_i requires carefully balancing the added term. Making c_i too small would not affect or only delay the numerical instability, while making them too large would introduce new numerical errors which would again cause numerical instability.

I found that $c_1 \neq 0$ leads to instabilities, which only leaves c_2 and c_3. I found that $c_2 = c_3 = c$ provided improved stability. To show the difference caused by changing the damping term, I compare a simulation with $c = 0$ to one using $c = 50$. Since the effect is purely in the inner layer, I only plot that section of the domain and set the vertical axis to match the magnitudes near the horizon. Figure 5.1 plots time slices of $h^{(5)}$ for both simulations. Introducing the extra factor certainly delays the onset of numerical instability. Increasing the resolution does not change the behavior for $h^{(5)}$, though it does for $h^{(8)}$. The effect of improving the numerical resolution is shown in Figure 5.2. While the increased resolution delayed the onset of the instability for $30M$, it did not resolve it. Hence, this problem is not primarily caused by insufficient numerical resolution and is instead coming from an issue with the evolution equations themselves.

Since the introduction of the c factors improves the numerical stability, I ran several different choices of c to test its effects. I tested with $c \in \{50, 100, 500, 1000, 10000\}$. For all of these simulations, I set gaussian parameters to $A = 1$, $\sigma = 2$, and $r_0 = 24.394449154672440$. The runtime parameter *r_center* is in Schwarzschild coordinates, and the value of r_0 comes from my choice of *r_center* $= 20$. I also set $S_- = -20$, which automatically sets $S_+ = 68.788898309344887$. The layer interfaces are at $T_- = -3.3520815669978354$ and $T_+ = 52.140979876342712$. For each choice of c, I ran simulations with DG orders of 8 and 10 and element counts of 32 and 64. Since I am simply examining the stability, I only simulate the $\ell = 2$ mode. I only show data for $m = 0$ because the evolution equations and initial data are independent of m, and the plots are identical for different m. As there are minor differences

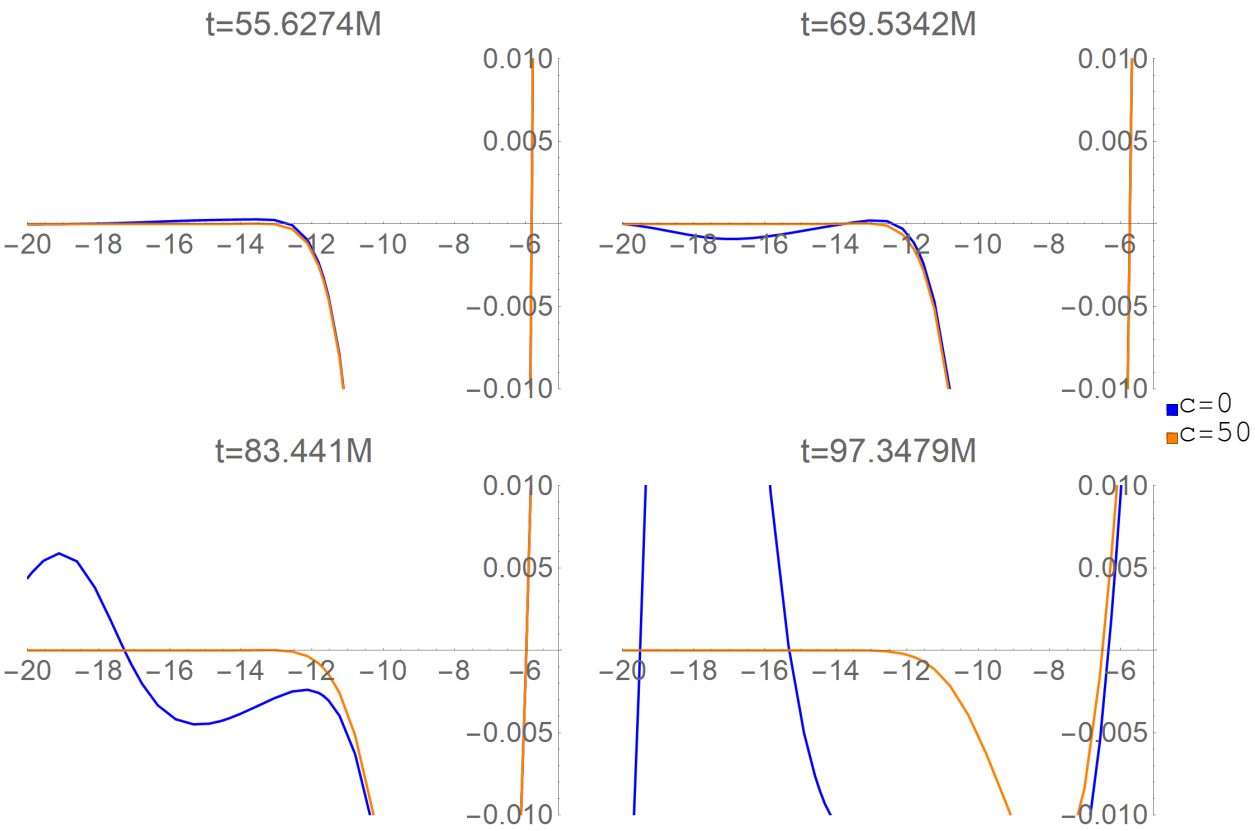

Figure 5.1. Comparison between simulations with $c = 0$ and $c = 50$. Several time slices of the $h^{(5)}$ field are shown, and the radial domain is restricted to the hyperboloidal region. The DG resolution was 32 elements at 8th order. The additional factor clearly counteracts the instability in the inner region.

between constraint-satisfying and constraint-violating initial data, I include plots of both. Figure 5.3 shows two time slices for the constraint-violating initial data with 32 elements, and Figure 5.4 gives the same plots with 64 elements. Similarly, Figure 5.5 shows two time slices for the constraint-satisfying initial data with 32 elements, and Figure 5.6 gives the same plots with 64 elements. In all four plots, the top row shows the data with DG order 8, and the bottom row shows the data with DG order 10. The most obvious feature is that in the first time slice, the data matches for all choices of c with constraint-satisfying initial data and almost matches for the constraint-violating initial data. Since the damping effect is stronger with higher c, the constraint-violating initial data unsurprisingly has some minor differences as the evolution is forced to be constraint-satisfying more quickly via the damping. Also,

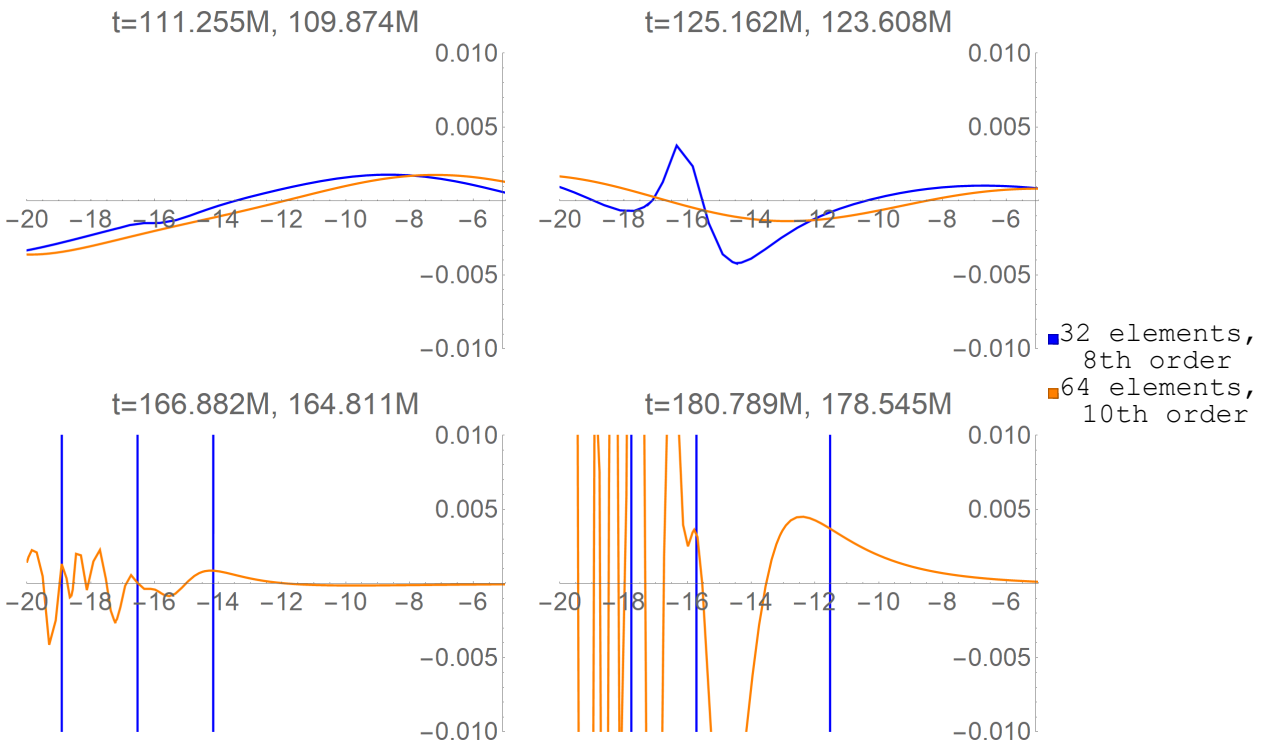

Figure 5.2. Plots showing the effects of increasing the DG resolution on the numerical instability for $h^{(8)}$ with $c = 0$. Two different times are given for each plot as the timestepping depends on the DG resolution. While the higher resolution simulation remains stable for slightly longer, the increase in resolution does not solve the core issue of the instability.

the second time slice shows that some choices of c are unstable. The simulation with 32 elements and order 8 is unstable with $c = 1000$, and the simulations with 64 elements are unstable with $c = 50$. To be able to easily plot any data, I replaced NaNs with 10^{133} so that all data was plottable. This is why the 32 element, order 8 simulation is a straight line at the top of the plot. It is entirely NaNs by $t = 152M$. The final feature is the noise that appears in several places in the domain. Specifically, these can be seen in the second time slices in Figures 5.3 and 5.5. The noise is present in just a single element: the first element in the hyperboloidal layer. The transition between the coordinates clearly introduces some numerical error. This feature is absent in Figures 5.4 and 5.6 because the error is reduced with increasing resolution. Figure 5.5 also has some noise in the element containing the center of the gaussian pulse. The reason this is visible in the constraint-satisfying initial data and not the constraint-violating initial data is likely due to the form of the initial data

for $h^{(i)}$, $i \in \{4, 7, 10\}$. This is particularly true of $h^{(7)}$, which is the derivative of a gaussian that peaks at around 500. Since the value is much larger, the numerical error is also larger, making it more visible in these plots.

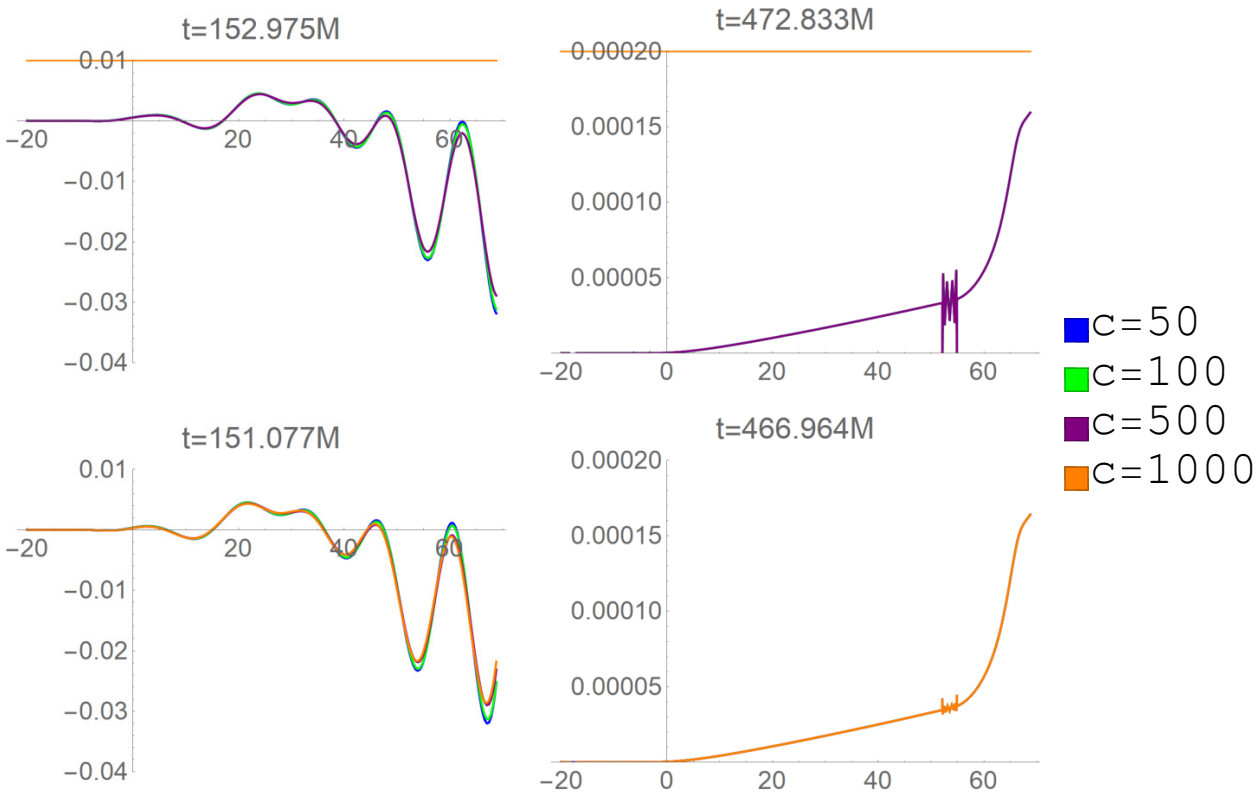

Figure 5.3. Plots of $h^{(5)}$ using constraint-violating initial data with 32 DG elements at two different time slices. The DG order 8 (top) and order 10 (bottom) show the same behavior. Minor differences are visible for the different choices of constraint, but the overall decay and oscillations match. The exception is $c = 1000$ at order 8, which is very unstable has become entirely NaN's by the first time slice.

Figures 5.3–5.6 show that the best choices for c are on the order of 100, but these simulations were only run to $t = 600M$. The initial data has mostly decayed to zero by that point, but numerical instability could arise later. Since actual self-force simulations will need to run longer, I ran simulations for $c \in \{100, 500\}$ up to $t = 10000$. For these simulations, I only used constraint-satisfying initial data. Figure 5.7 shows the final outputted time slice for $h^{(5)}$ for these simulations. The scalar field has almost completely decayed away, being on the order of 10^{-6}. The reduction of numerical error at the layer interfaces and the source clearly

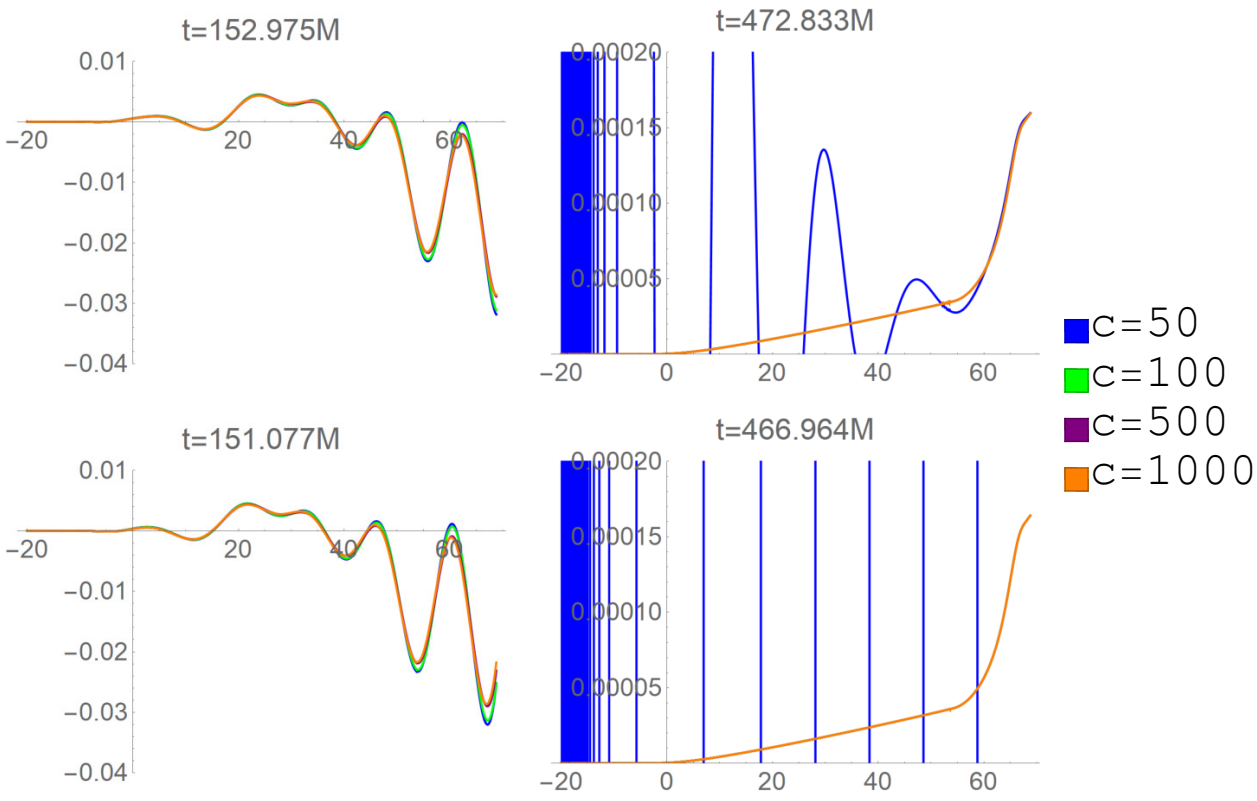

Figure 5.4. Plots of $h^{(5)}$ using constraint-violating initial data with 64 DG elements at two different time slices. The DG order 8 (top) and order 10 (bottom) show the same behavior. Minor differences are visible for the different choices of constraint, but the overall decay and oscillations match. The exception is $c = 50$, which is very unstable and grows uncontrollably at later times.

decreases with increasing DG resolution, as expected. However, the $c = 100$ simulation with 64 elements at 10th order is again unstable.

To summarize, the 32 element, 8th order simulation becomes unstable with $c = 1000$. On the other hand, the 64 element simulations are unstable with $c = 50$. Finally, the 64 element, 10th order simulation is unstable with $c = 100$ at long times. From this, a general trend becomes clear. As the DG resolution increases, higher c values are more stable, and vice versa. Of all the tested values, constraint damping with $c = 500$ is stable for all tested DG configurations and match the results from other stable choices of c for a given DG resolution. Hence, I will restrict any further analysis to the $c = 500$ simulations.

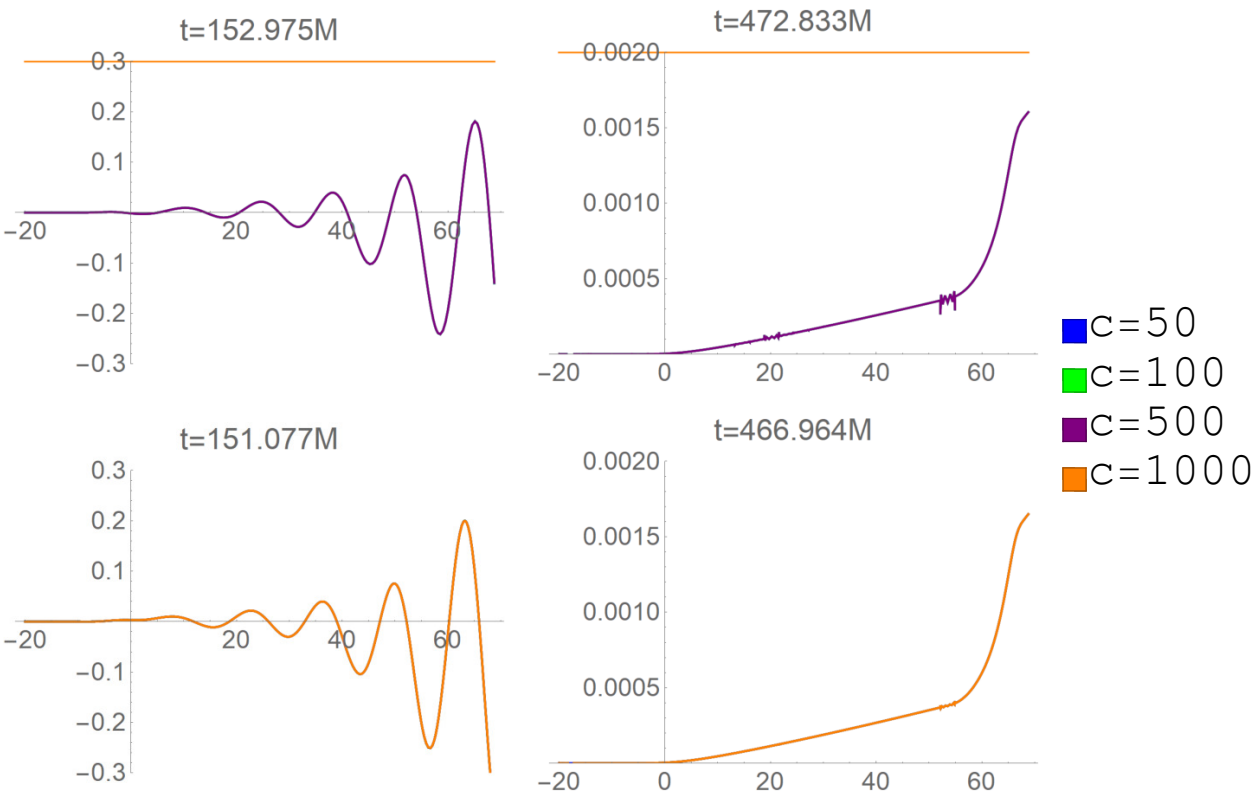

Figure 5.5. Plots of $h^{(5)}$ using constraint-satisfying initial data with 32 DG elements at two different time slices. The DG order 8 (top) and order 10 (bottom) show the same behavior. The evolution perfectly matches for all choices of c except $c = 1000$ at order 8, which is very unstable has become entirely NaNs by the first time slice.

5.4.2. Constraint Damping

Now that I have chosen c, the next important criterion for validating the code is examining the constraints during the evolution. For the constraint-violating initial data, the expectation is that the constraints will quickly fall to zero once the initial wave has propagated out. Since the initial data is 'bad', the constraint damping is limited in its effect until the initial data has propagated out of the domain. For the constraint-satisfying initial data, the ideal result would be that the constraints remain zero for the entire simulation. However, numerical error will always introduce some violations even with perfect initial data, and the damping is meant to quickly correct any deviations from zero. For the constraint analysis, I only show the results for the simulations using 64 elements and 10th order, as the increasing order does not directly improve the constraint damping.

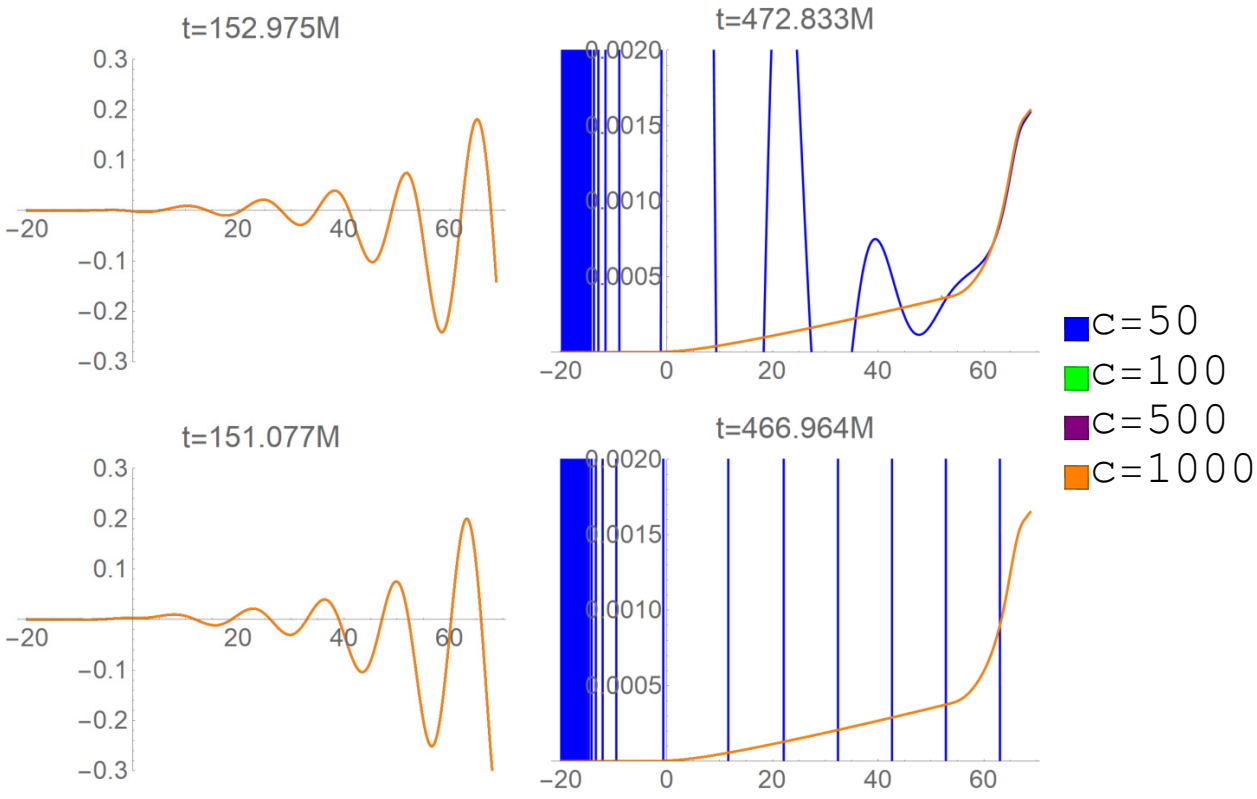

Figure 5.6. Plots of $h^{(5)}$ using constraint-satisfying initial data with 64 DG elements at two different time slices. The DG order 8 (top) and order 10 (bottom) show the same behavior. The evolution perfectly matches for all choices of c except $c = 50$, which is unstable and grows uncontrollably during the simulation.

Figure 5.8 shows the first four outputted time slices for the constraints. At time $t = 0$, all the constraints are zero, as expected. The odd sector constraint remains zero, but the three even sector constraints do not. The reason for this is, once again, the initial data for $h^{(7)}$. While analytically the data is constraint-satisfying, $h^{(7)}$ and its derivatives are significantly larger than the other fields, and any calculations involving differences between them will naturally introduce numerical errors. Due to this, constraint violations arise in the simulation despite the (analytically) constraint-satisfying initial data. If a better formulation of initial data were used, the even sector constraints should behave similarly to the odd sector constraint. Lower resolution simulations primarily differ from these results in the odd sector where the constraint violations arise from numerical error in the calculations and not from a difference in magnitude of the fields themselves. Because of this, the constraint violations

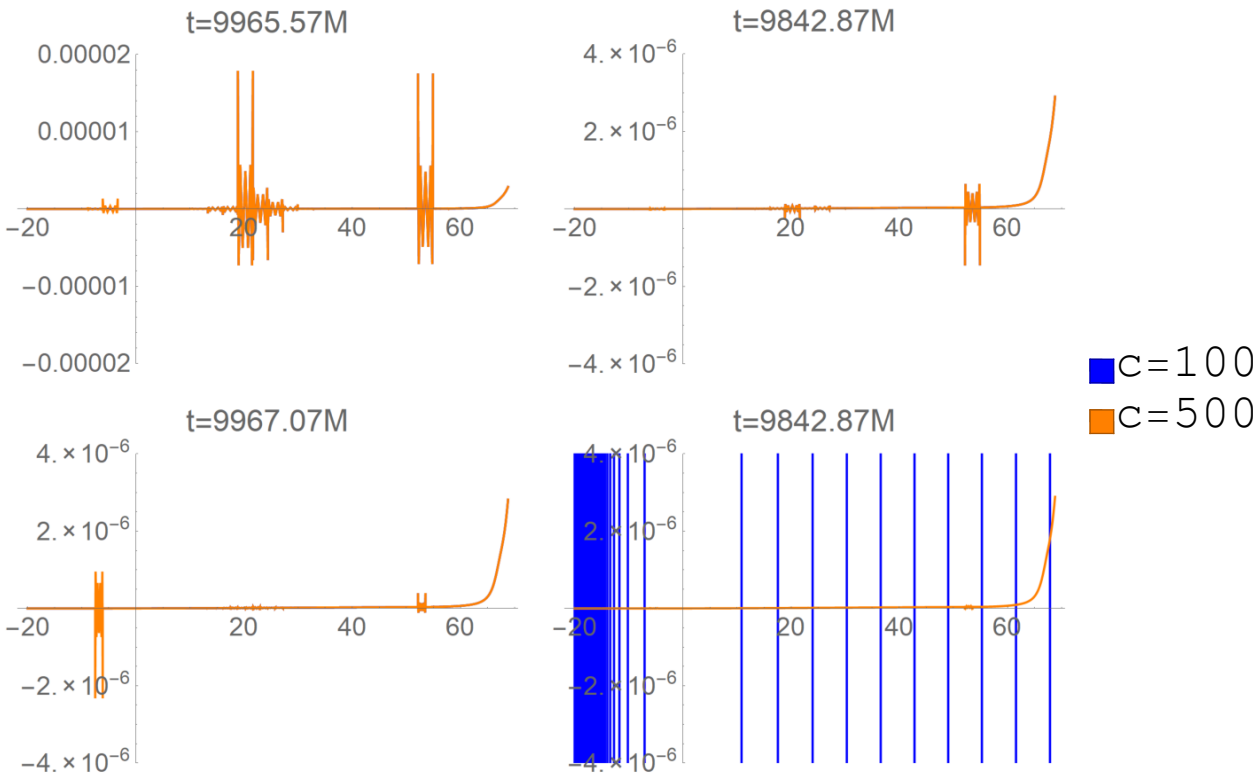

Figure 5.7. Plots of $h^{(5)}$ using constraint-satisfying initial data at the final outputted time slice. The top row uses 32 DG elements, and the bottom row uses 64 DG elements. The left column uses 8th order, and the right column uses 10th order. Even at long times, $c = 500$ is stable for all the tested configurations. However, $c = 100$ is unstable at the highest resolution.

at the interface in Z_4 are three orders of magnitude lower in this run than in the lowest resolution simulation. Figure 5.9 shows the same simulation later in the run. The constraint violations approach, and Z_4 has already reached round-off error.

For comparison, Figures 5.10 and 5.11 show the same plots for the constraint-violating initial data. As expected, the constraints are initially non-zero and resemble the gaussian pulses I used as initial data. Since the even sector of both cases have similar magnitudes shortly after the simulation starts, the decay rate of Z_1, Z_2, and Z_3 are the same. Interestingly, the odd sector constraint Z_4 decays much faster than the other three, having reached 10^{-12} at $t = 260M$. This behavior is not evident with constraint-satisfying initial data because Z_4 starts with a much lower magnitude in that simulation.

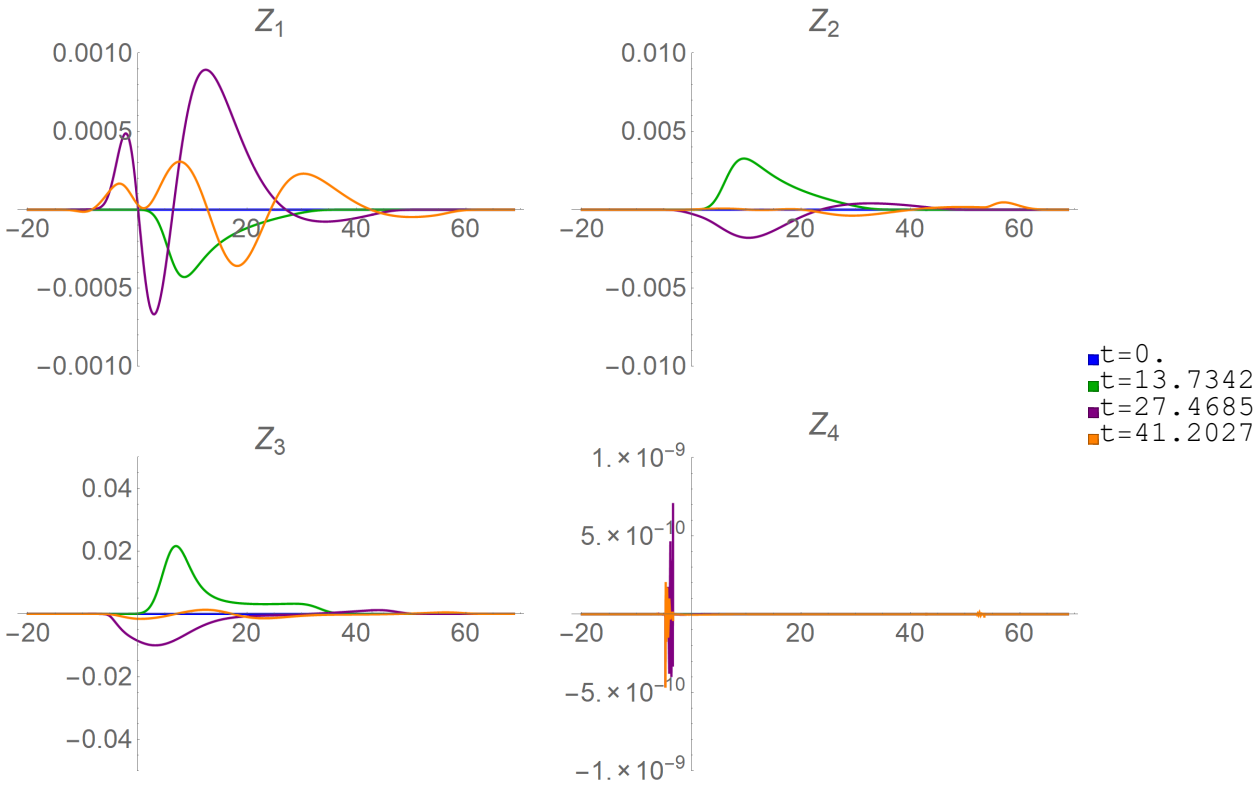

Figure 5.8. Plots the constraints with constraint-satisfying initial data. At $t = 0$, the constraints are zero. The constraints become excited by numerical errors resulting from the difference in magnitude of the fields, but the magnitudes of the constraint violations die off over time.

5.4.3. Metric Reconstruction

Figures 5.8–5.11 display the constraint damping present in the code. The best example of its effect is in the odd sector, as that domain is not affected by the large difference in magnitude between $h^{(7)}$ and the other scalar fields. A better assessment of these effects would require a new formulation for the initial data, which is not currently available. As such, the next stage of assessing the code is examining the behavior of the metric perturbation. First, the metric must be reconstructed from the $h^{(i)}$. The linear combinations of the scalar fields to give the tensor components of the metric pertrubation are given by Barack and Lousto [56].

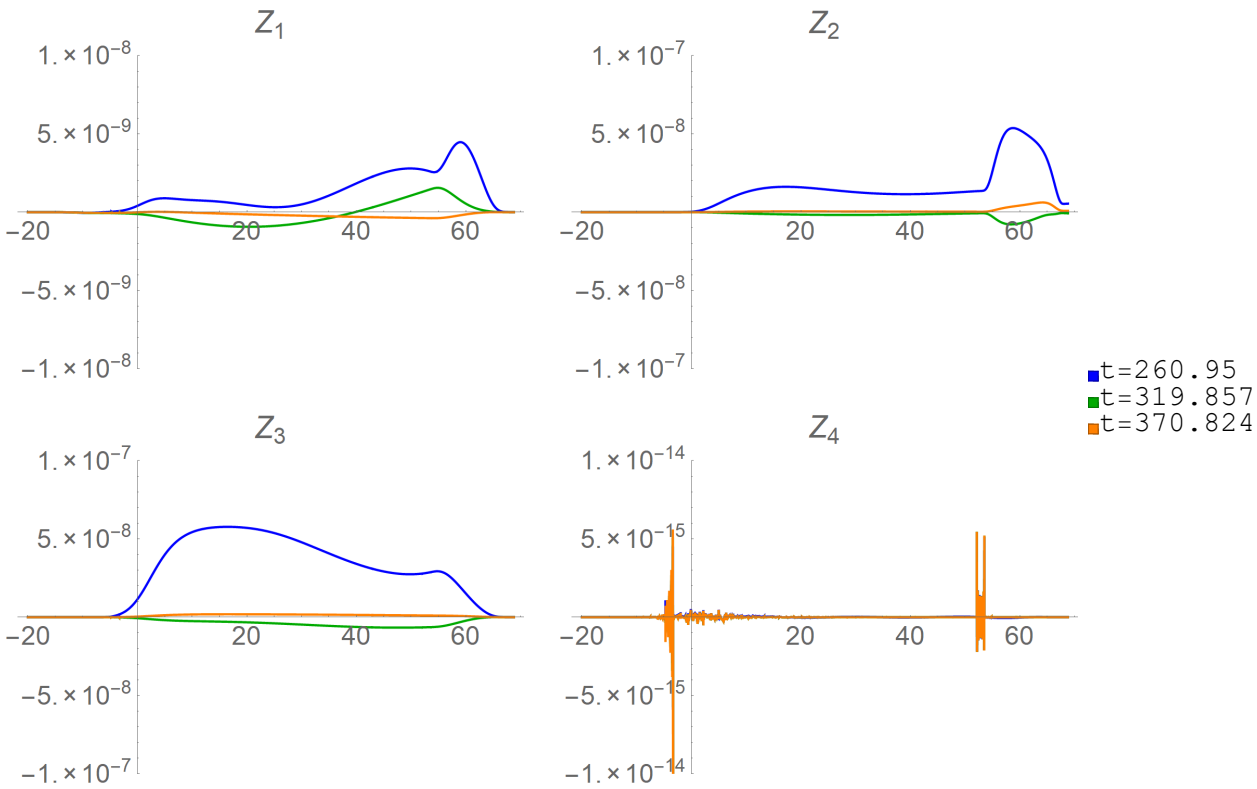

Figure 5.9. Plots of the constraints with constraint-satisfying initial data. By $t = 260M$, Z_4 is at round-off error, and it remains at this order for the remainder of the simulation.

As my basis is slightly different from theirs, the metric reconstruction equations are

$$h^{\ell m}_{tt} = \left(\overline{h}^{(1)\ell m} + \overline{h}^{(2)\ell m} + f\overline{h}^{(6)\ell m}\right) Y^{\ell m} \tag{5.20}$$

$$h^{\ell m}_{tr_*} = f^{-1}\left(\overline{h}^{(1)\ell m} - \overline{h}^{(2)\ell m}\right) Y^{\ell m} \tag{5.21}$$

$$h^{\ell m}_{r_*r_*} = f^{-2}\left(\overline{h}^{(1)\ell m} + \overline{h}^{(2)\ell m} - f\overline{h}^{(6)\ell m}\right) Y^{\ell m} \tag{5.22}$$

$$h^{\ell m}_{t\theta} = r\left[\left(\overline{h}^{(4)} + \overline{h}^{(5)}\right) Y^{\ell m}_{V1} + \left(\overline{h}^{(8)} + \overline{h}^{(9)}\right) Y^{\ell m}_{V2}\right] \tag{5.23}$$

$$h^{\ell m}_{t\phi} = r\sin(\theta)\left[\left(\overline{h}^{(4)} + \overline{h}^{(5)}\right) Y^{\ell m}_{V2} - \left(\overline{h}^{(8)} + \overline{h}^{(9)}\right) Y^{\ell m}_{V1}\right] \tag{5.24}$$

$$h^{\ell m}_{r_*\theta} = r\left[\left(\overline{h}^{(4)} - \overline{h}^{(5)}\right) Y^{\ell m}_{V1} + \left(\overline{h}^{(8)} - \overline{h}^{(9)}\right) Y^{\ell m}_{V2}\right] \tag{5.25}$$

$$h^{\ell m}_{r_*\phi} = r\sin(\theta)\left[\left(\overline{h}^{(4)} - \overline{h}^{(5)}\right) Y^{\ell m}_{V2} - \left(\overline{h}^{(8)} - \overline{h}^{(9)}\right) Y^{\ell m}_{V1}\right] \tag{5.26}$$

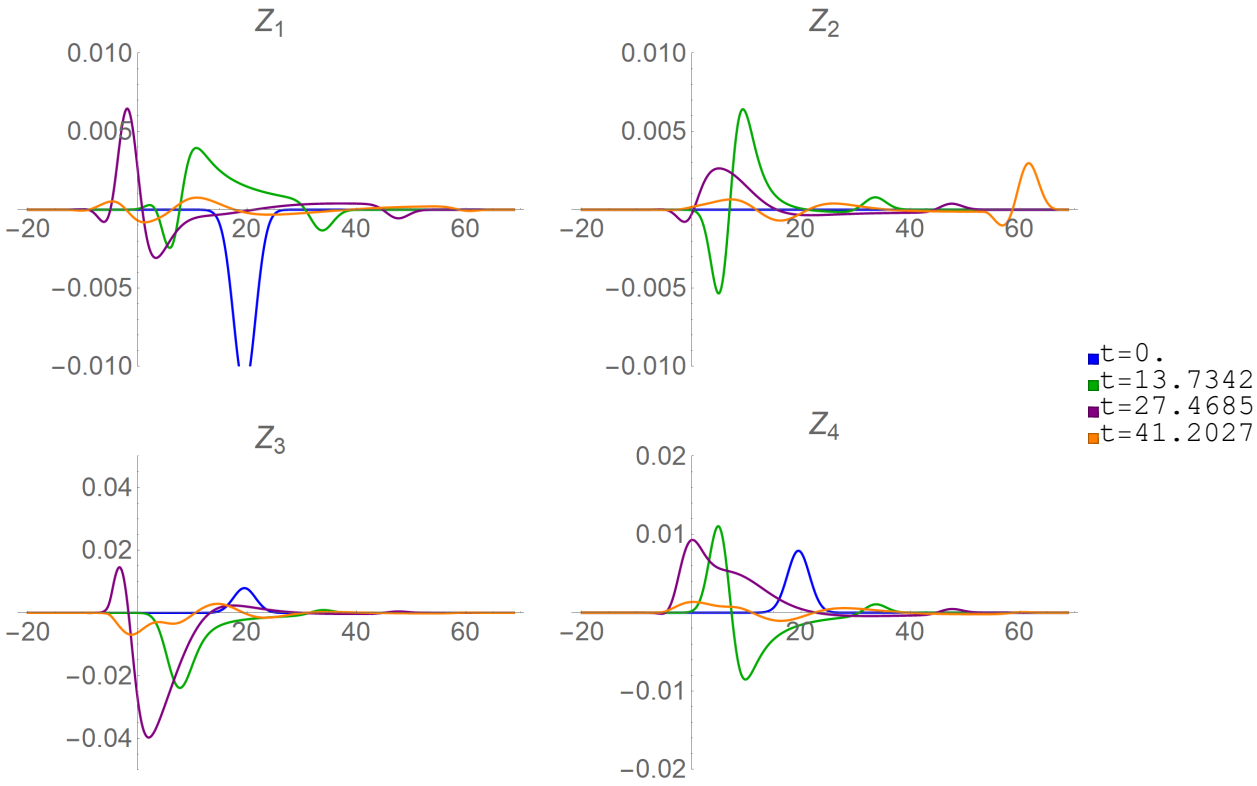

Figure 5.10. Plots of the constraints with constraint-violating initial data. At $t = 0$, the constraints are non-zero, as expected.

$$h^{\ell m}_{\theta\theta} = r^2 \left(\overline{h}^{(3)} Y^{\ell m} + \overline{h}^{(7)} Y^{\ell m}_{T1} + \overline{h}^{(10)} Y^{\ell m}_{T2} \right) \tag{5.27}$$

$$h^{\ell m}_{\theta\phi} = r^2 \sin(\theta) \left(\overline{h}^{(7)} Y^{\ell m}_{T2} - \overline{h}^{(10)} Y^{\ell m}_{T1} \right) \tag{5.28}$$

$$h^{\ell m}_{\phi\phi} = r^2 \sin^2(\theta) \left(\overline{h}^{(3)} Y^{\ell m} - \overline{h}^{(7)} Y^{\ell m}_{T1} - \overline{h}^{(10)} Y^{\ell m}_{T2} \right) \tag{5.29}$$

where

$$Y^{\ell m}_{V1} = \frac{1}{\lambda_1^2} \partial_\theta Y^{\ell m}$$

$$Y^{\ell m}_{V2} = \frac{1}{\lambda_1^2 \sin(\theta)} \partial_\phi Y^{\ell m}$$

$$Y^{\ell m}_{T1} = \frac{1}{\lambda_2^2} \left[\sin(\theta) \partial_\theta \left(\sin^{-1}(\theta) \partial_\theta Y^{\ell m} \right) - \sin^2(\theta) \partial_\phi^2 Y^{\ell m} \right]$$

$$Y^{\ell m}_{T2} = \frac{2}{\lambda_2^2} \partial_\theta \left(\sin^{-1}(\theta) \partial_\phi Y^{\ell m} \right)$$

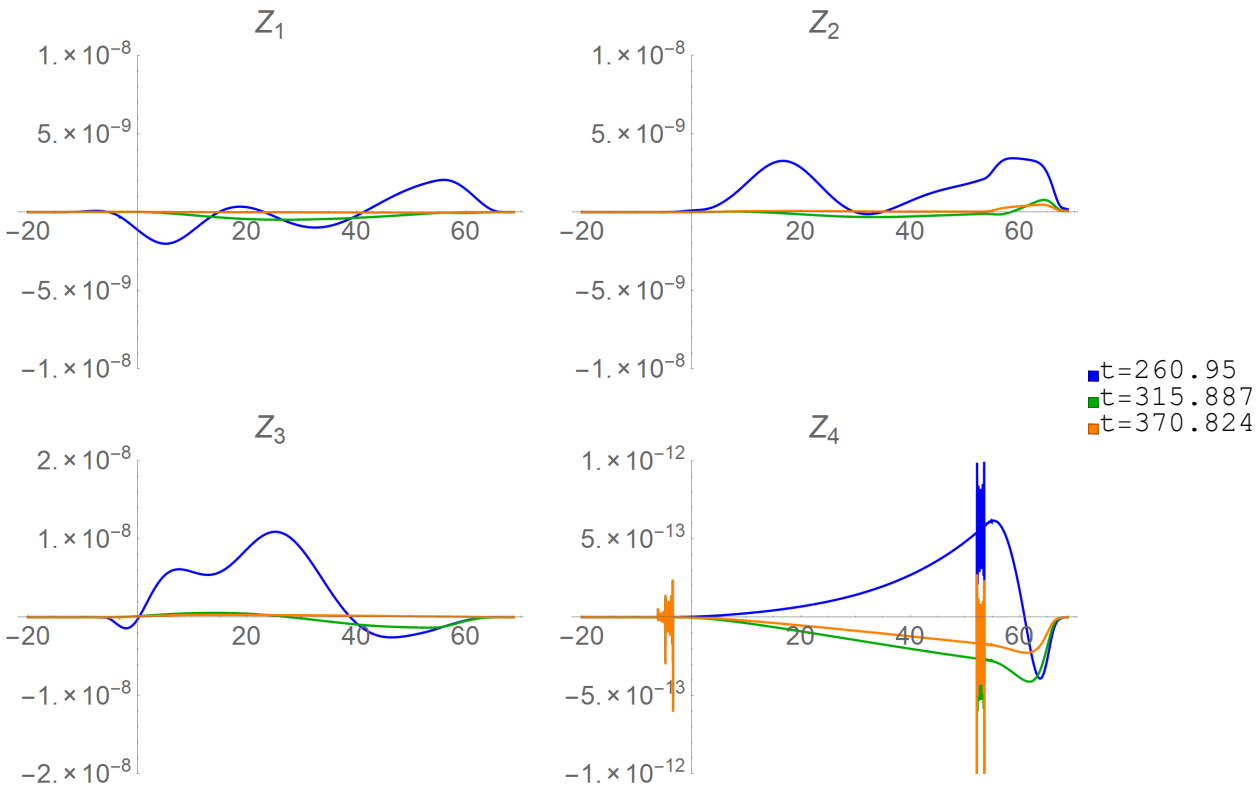

Figure 5.11. Plots of the constraints with constraint-violating initial data. While Z_4 started with the same magnitude as the other constraints, it damps far more quickly.

I use here the same notation as in Appendix A with $\lambda_1^2 \equiv \ell(\ell+1)$ and $\lambda_2^2 \equiv \lambda_1^2(\ell-1)(\ell+2)$.

The final test of the validity of the code is whether the behavior of the metric reconstruction matches the quasinormal mode ringdown behavior. If the QNM frequencies are correct, then the code is ready for introducing the effective source and moving on to circular orbits. To extract these frequencies, I output data for the reconstructed metric perturbation at a single radial location and fit that data with a damped sinusoid. The expected form of the QNM ringdown is of the form

$$h^\ell = Ae^{\omega_I t}\sin(\omega_R t) \tag{5.30}$$

The $\ell = 2$ QNM frequency for a Schwarzschild black hole is

$$\omega_R = 0.37367168441804177 \tag{5.31}$$

$$\omega_I = -0.08896231568893546 \tag{5.32}$$

I used the *QuasiNormalModes* Mathematica package from the Black Hole Perturbation Toolkit to generate these values [60].

Unfortunately, the QNM frequencies are incorrect for all the data I have analyzed. Changes in DG resolution also do not improve the results. To verify that I was comparing to the correct frequencies, I used the *qnm* Python package[61] from the Black Hole Perturbation Toolkit to verify the values in Equations 5.31 and 5.32. Normally, to extract the frequencies, I plot the logarithm and quickly extract the frequency. However, the reconstructed metric is clearly not just a damped sinusoid, as shown in Figure 5.12. The lowest and highest resolutions match, so the solution is also not converging to a different answer.

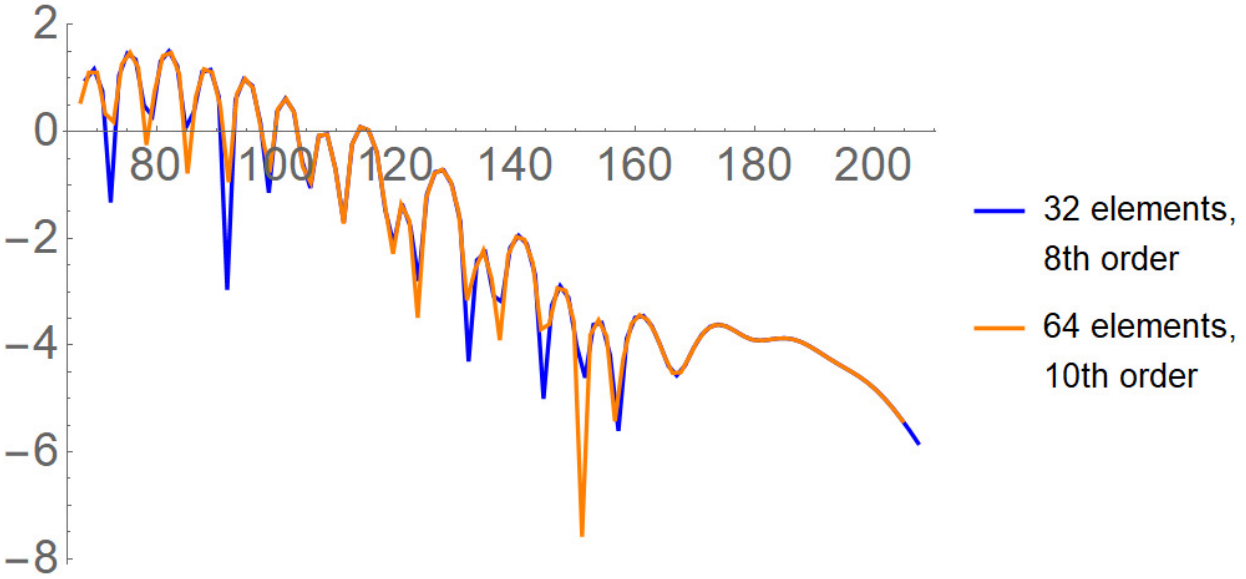

Figure 5.12. Plot of the log of the $h_{r\phi}$ component of the metric perturbation. The data is not a damped sinusoid.

While the data clearly fails to capture the ringdown behavior, I do attempt to extract an approximate frequency by plotting a damped sinusoid over the data. Since the data does not have pure damped sinusoid behavior, I perform an approximate fit by eye on $\log(\text{abs}(h_{r\phi}))$. This fit is shown in Figure 5.13. I do this because the time series data appears like it might be a combination of multiple sinusoids. As such, a more 'correct' fit is unlikely to give any meaningful information until the reason for this behavior is found and fixed. I give

the frequencies from my fit and from frequency domain calculations in Table 5.3. While a fit by eye is very approximate, it is sufficient to show that my data does not have the proper frequency or decay rate. The reasons for the large differences in the frequencies is

	Time domain results	Frequency domain solution
ω_R	0.47	0.374
ω_I	−0.06	−0.0890

Table 5.3. Comparison of QNM frequencies between my code and frequency domain results. Frequencies for my data were determined by adjusting the sinusoid parameters until it most closely matched the numerical data.

currently unknown, but it is possible the constraint violations are polluting the ringdown effect. Alternatively, the increased numerical errors in the first element in each hyperboloidal layer could be causing artificial waves to propagate back into the tortoise region where this extraction occurs.

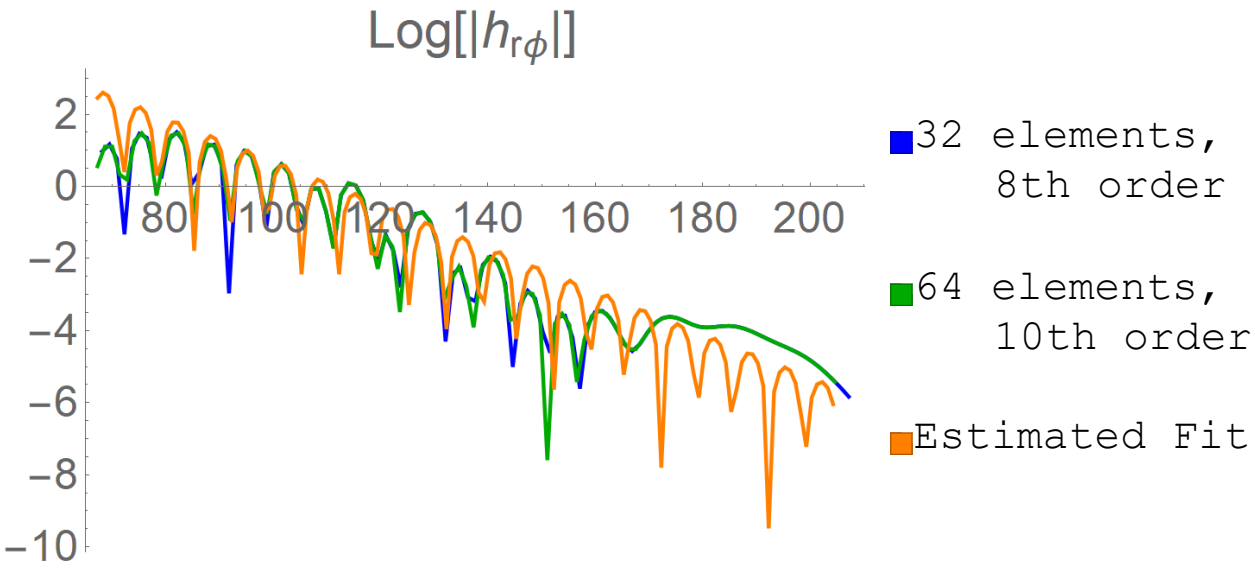

Figure 5.13. Plot of the logarithm of the $h_{r\phi}$ component of the metric perturbation. A fit estimated by eye has been added to approximate the dominant frequency behavior of the data.

This code is still in development, but significant progress has been made towards having a functional Lorenz gauge code. Issues with stability have been addressed through numerical experimentation, and the constraint damping successfully damps any constraint violations.

The automated Mathematica notebooks also allow for quick generation of new versions of the code with different choices of basis and constraint damping. Future work will focus on ruling out possible causes for the data to not behave as a pure damped sinusoid.

Chapter 6. Conclusion

My dissertation work consists of two components. First, I contributed to the core of the Einstein Toolkit by improving the Carpet AMR driver. Second, I wrote code to simulate the gravitational self-force in two different gauges. In the Regge-Wheeler-Zerilli gauge my code successfully simulates circular orbits. In the Lorenz gauge my code is stable and produces waves which propagate out of the domain, marking significant progress towards having a code which can handle the significantly more complicated evolution equations of the Lorenz gauge.

The original Cactus system required thorn writers to explicitly specify when ghost zone synchronization and the application of boundary conditions would occur for each grid function. This method places unnecessary burden on contributors who may not be as familiar with Cactus' method of scheduling subroutines. It also obscures the efficiency of the code, as triggering syncs too often is a serious performance issue that could not be easily detected. I replaced this system with PreSync which instead relies on the concept of 'reads' and 'writes'. The scheduling of each subroutine contains information about what grid functions it accesses, which is purely local information that does not depend on anything outside that subroutine. This removes the need for thorn writers to know where the subroutines will be in Cactus' schedule tree. They need only schedule the subroutines and provide the 'read'/'write' information. The PreSync-enabled Cactus uses this information to automatically trigger ghost zones synchronization and the application of boundary conditions as required. By automating the process, the risk of over-synchronization is also greatly reduced. In addition, I added features to assist in error-checking while using PreSync. New macros only provide access to the grid functions declared in the schedule.ccl instead of all grid functions. The poisoning feature writes NaNs into the data when the grid function is declared as 'write' to prevent the data from being accidentally used without an associated 'read' declaration. The `ReadWriteDiagnostics` thorn provides feedback on the suggested region of validity for declared grid functions. All these features improve the user experience as well as enhance the

ETK's computational efficiency. This work is published as a technical paper at the PEARC Conference [2] and has been merged into the master branch of the Einstein Toolkit. I am also currently working on the CarpetX driver, which is a successor to Carpet that is built from the ground up with support for PreSync. This will encourage adoption of the new scheduling paradigm and help transition the ETK community towards learning and using the new features of PreSync.

My contributions to the self-force community consist of two new codes, both for simulating the gravitational self-force in the time domain. They take advantage of many techniques and tricks—such as Discontinuous Galerkin method, hyperboloidal foliation, and the effective source approach—to make a fully self-consistent evolution of the EMRI system possible. The code in the RWZ gauge successfully simulates circular orbits and rapidly converges to frequency domain results. The code is also ready for simulating eccentric orbits, and the initial data code is capable of generating eccentric initial data and results for comparison. Once an effective source code is available, the code can be immediately compared to frequency domain results for eccentric orbits. I am in the process of publishing a peer-reviewed paper with these results, and the code is currently in review for acceptance into the Einstein Toolkit. The code will also be added to the Black Hole Perturbation Toolkit, which is a collection of computational codes for the EMRI community.

The Lorenz gauge code is less developed than the RWZ code, but significant progress has been made. Finding a basis for which the evolution equations could be finite at both the horizon and infinity proved difficult, as did ensuring that the methods employed to do so preserved the constraint damping. The current version of the code is numerically stable for long times and rapidly damps any constraint violations. While the QNM frequencies do not match and the data shows some numerical artifacts, I have developed a streamlined system for deriving and implementing new variants of the code. These Mathematica notebooks simplify the process of revising the code and allow for quick and easy changes to constraint damping, choice of basis, and other parameters without the burden of retyping the lengthy

code by hand. The extra effort required for the Lorenz code is also justified, as second-order self-force is currently being developed in the Lorenz gauge. As second-order accuracy is required to meet LISA's needs, working in the more difficult gauge is likely necessary to prepare for the eventual extension to second-order.

My dissertation research serves to improve the computational tools available to the gravitational community, both in the established Einstein Toolkit community, and the newer Black Hole Perturbation Toolkit community. The self-force codes represent one of the first fully self-consistent time domain codes for the gravitational self-force.

Appendix A. Tensor Spherical Harmonic Basis

The tensor spherical harmonic basis $Y^{(i)\ell m}_{\alpha\beta}$ can be chosen in a variety of ways. Here, I give the basis used by Barack and Sago[57] in Schwarzschild coordinates $\{t, r, \theta\phi\}$. Note that $Y^{(3)\ell m}_{\alpha\beta}$ differs from the basis used by Barack and Lousto [56] by a factor of f. This was introduced to improve the behavior of the evolution equations at the horizon. While the perturbation equations are in coordinates other than Schwarzschild, I use this tensor spherical harmonic basis and then explicitly transform the basis components in the Mathematica notebook. For the following, $Y^{\ell m}$ are the standard spherical harmonics, $s \equiv \sin(\theta)$, $f \equiv 1 - \frac{2M}{r}$, $\lambda_1^2 \equiv \ell(\ell+1)$, and $\lambda_2^2 \equiv \lambda_1^2(\ell-1)(\ell+2)$. The angular operators D_1 and D_2 are given by

$$D_1 \equiv 2\partial_{\theta\phi} - 2\cot(\theta)\partial_\phi, \;\; D_2 \equiv \partial_{\theta\theta} - \cot(\theta)\partial_\theta - s^{-2}\partial_{\phi\phi} \tag{A.1}$$

With these definitions, the $Y^{(i)\ell m}_{\alpha\beta}$ are given by

$$Y^{(1)\ell m}_{\alpha\beta} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & f^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}, \qquad Y^{(2)\ell m}_{\alpha\beta} = \tfrac{1}{f\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}$$

$$Y^{(3)\ell m}_{\alpha\beta} = \tfrac{f}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -f^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}, \qquad Y^{(4)\ell m}_{\alpha\beta} = \tfrac{r}{\lambda_1\sqrt{2}} \begin{pmatrix} 0 & 0 & \partial_\theta & \partial_\phi \\ 0 & 0 & 0 & 0 \\ \partial_\theta & 0 & 0 & 0 \\ \partial_\phi & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}$$

$$Y^{(5)\ell m}_{\alpha\beta} = \tfrac{r}{f\lambda_1\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \partial_\theta & \partial_\phi \\ \partial_\theta & 0 & 0 & 0 \\ \partial_\phi & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}, \qquad Y^{(6)\ell m}_{\alpha\beta} = \tfrac{r^2}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s^2 \end{pmatrix} Y^{\ell m}$$

$$Y^{(7)\ell m}_{\alpha\beta} = \tfrac{r^2}{\lambda_2\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & D_2 & D_1 \\ 0 & 0 & D_1 & -s^2 D_2 \end{pmatrix} Y^{\ell m}$$

$$Y^{(8)\ell m}_{\alpha\beta} = \tfrac{r}{f\lambda_1\sqrt{2}} \begin{pmatrix} 0 & 0 & s^{-1}\partial_\phi & -s\partial_\theta \\ 0 & 0 & 0 & 0 \\ s^{-1}\partial_\phi & 0 & 0 & 0 \\ -s\partial_\theta & 0 & 0 & 0 \end{pmatrix} Y^{\ell m}$$

$$Y^{(9)\ell m}_{\alpha\beta} = \frac{r}{f\lambda_1\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & s^{-1}\partial_\phi & -s\partial_\theta \\ 0 & s^{-1}\partial_\phi & 0 & 0 \\ 0 & -s\partial_\theta & 0 & 0 \end{pmatrix} Y^{\ell m}$$

$$Y^{(10)\ell m}_{\alpha\beta} = \frac{r^2}{\lambda_2\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & s^{-1}D_1 & -sD_2 \\ 0 & 0 & -sD_2 & -sD_1 \end{pmatrix} Y^{\ell m}$$

Appendix B. Separation of the Evolution Equations

To evolve the gravitational self-force, I must first find separable equations for each field $h^{(i)}$. This means choosing a combination of the equations (and derivatives of equations) resulting from substituting Equation 5.3 into Equation 5.2 which only has the principal part for a single $h^{(i)}$. The combinations for tortoise coordinates are given in Table B.1. The combinations for the inner and outer hyperboloidal regions are given in Tables B.2 and B.3, respectively. Q refers to the tensor components of the perturbation equation. For example, $Q_{tt} + Q_{r_*r_*}$ means to add the t, t and r_*, r_* components of Equation 5.2. As in Appendix A, I use the definitions

$$s \equiv \sin(\theta),\ D_1 \equiv 2\partial_{\theta\phi} - 2\cot(\theta)\partial_\phi,\ D_2 \equiv \partial_{\theta\theta} - \cot(\theta)\partial_\theta - s^{-2}\partial_{\phi\phi} \tag{B.1}$$

Note that changing from the inner and outer layers only has two steps. First, the signs $1 + H \leftrightarrow 1 - H$ are swapped. This is unsurprising given the similarity of the two coordinates. The other change is the equations with H terms change Specifically, the equations for $\{1, 2\}$, $\{4, 5\}$, and $\{8, 9\}$ are all exchanged. This comes from the transformation used to generate my basis. The linear combinations chosen separate these field pairs into ingoing and outgoing fields. Since each layer preserves one of these types of waves, that equation remains unchanged from the tortoise region. Thus, the conversion from the inner to outer layer amounts to switching the equations between these pairs and changing the signs of all the $1 \pm H$ terms. Also, it is clear that the equations in the hyperboloidal layers reduce to those of in the tortoise region in the limit $H \to 0$, as one would expect. Some of the multiplicative factors are singular in the limit towards the horizon and $\mathscr{I}^+$, but recall that these are not the final evolution equations. Once these combinations are constructed, the second time derivative of the $h^{(i)}$ must be isolated on one side of the equation, and the other side will be divided by its coefficient. Thus, one can multiply these combinations by any factor and still arrive at the same final equation.

Table B.1. Explicit combinations to construct the evolution equations in tortoise coordinates $\{t, r_*, \theta, \phi\}$.

To get the equation for...	use the combination...
$i = 1$	$Q_{tt} + 2Q_{tr_*} + Q_{r_*r_*}$
$i = 2$	$Q_{tr_*} - 2Q_{tr_*} + Q_{r_*r_*}$
$i = 3$	$Q_{tt} - Q_{r_*r_*}$
$i = 4$	$\partial_\theta \left[s \left(Q_{t\theta} + Q_{r_*\theta} \right) \right] + \partial_\phi \left[s^{-1} \left(Q_{t\phi} + Q_{r_*\phi} \right) \right]$
$i = 5$	$\partial_\theta \left[s \left(Q_{t\theta} - Q_{r_*\theta} \right) \right] + \partial_\phi \left[s^{-1} \left(Q_{t\phi} - Q_{r_*\phi} \right) \right]$
$i = 6$	$Q_{\theta\theta} + s^{-2} Q_{\phi\phi}$
$i = 7$	$\frac{2}{s} D_1 \left(s Q_{\theta\phi} \right) + D_2 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi} \right)$
$i = 8$	$\partial_\phi \left(Q_{t\theta} + Q_{r_*\theta} \right) - \partial_\theta \left(Q_{t\phi} + Q_{r_*\phi} \right)$
$i = 9$	$\partial_\phi \left(Q_{t\theta} - Q_{r_*\theta} \right) - \partial_\theta \left(Q_{t\phi} - Q_{r_*\phi} \right)$
$i = 10$	$D_1 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi} \right) - 2s D_2 \left(s Q_{\theta\phi} \right)$

Table B.2. Explicit combinations to construct the evolution equations in hyperboloidal coordinates $\{\tau, \rho, \theta, \phi\}$ for the inner layer.

To get the equation for...	use the combination...
$i = 1$	$\frac{1-H}{1+H} Q_{\tau\tau} + 2Q_{\tau\rho} + \frac{1+H}{1-H} Q_{\rho\rho}$
$i = 2$	$Q_{\tau\tau} - 2Q_{\tau\rho} + Q_{\rho\rho}$
$i = 3$	$Q_{\tau\tau} + \frac{2H}{1-H} Q_{\tau\rho} - \frac{1+H}{1-H} Q_{\rho\rho}$
$i = 4$	$\partial_\theta \left[s \left(Q_{\tau\theta} + \frac{1+H}{1-H} Q_{\rho\theta} \right) \right] + \partial_\phi \left[s^{-1} \left(Q_{\tau\phi} + \frac{1+H}{1-H} Q_{\rho\phi} \right) \right]$
$i = 5$	$\partial_\theta \left[s \left(Q_{\tau\theta} - Q_{\rho\theta} \right) \right] + \partial_\phi \left[s^{-1} \left(Q_{\tau\phi} - Q_{\rho\phi} \right) \right]$
$i = 6$	$Q_{\theta\theta} + s^{-2} Q_{\phi\phi}$
$i = 7$	$\frac{2}{s} D_1 \left(s Q_{\theta\phi} \right) + D_2 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi} \right)$
$i = 8$	$\partial_\phi \left(Q_{\tau\theta} + \frac{1+H}{1-H} Q_{\rho\theta} \right) - \partial_\theta \left(Q_{\tau\phi} + \frac{1+H}{1-H} Q_{\rho\phi} \right)$
$i = 9$	$\partial_\phi \left(Q_{\tau\theta} - Q_{\rho\theta} \right) - \partial_\theta \left(Q_{\tau\phi} - Q_{\rho\phi} \right)$
$i = 10$	$D_1 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi} \right) - 2s D_2 \left(s Q_{\theta\phi} \right)$

Table B.3. Explicit combinations to construct the evolution equations in hyperboloidal coordinates $\{\tau, \rho, \theta, \phi\}$ for the outer layer.

To get the equation for…	use the combination…
$i = 1$	$Q_{\tau\tau} + 2Q_{\tau\rho} + Q_{\rho\rho}$
$i = 2$	$\frac{1+H}{1-H}Q_{\tau\tau} - 2Q_{\tau\rho} + \frac{1-H}{1+H}Q_{\rho\rho}$
$i = 3$	$Q_{\tau\tau} + \frac{2H}{1+H}Q_{\tau\rho} - \frac{1-H}{1+H}Q_{\rho\rho}$
$i = 4$	$\partial_\theta \left[s \left(Q_{\tau\theta} + Q_{\rho\theta}\right)\right] + \partial_\phi \left[s^{-1} \left(Q_{\tau\phi} + Q_{\rho\phi}\right)\right]$
$i = 5$	$\partial_\theta \left[s \left(Q_{\tau\theta} - \frac{1-H}{1+H}Q_{\rho\theta}\right)\right] + \partial_\phi \left[s^{-1} \left(Q_{\tau\phi} - \frac{1-H}{1+H}Q_{\rho\phi}\right)\right]$
$i = 6$	$Q_{\theta\theta} + s^{-2}Q_{\phi\phi}$
$i = 7$	$\frac{2}{s}D_1 \left(sQ_{\theta\phi}\right) + D_2 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi}\right)$
$i = 8$	$\partial_\phi \left(Q_{\tau\theta} + Q_{\rho\theta}\right) - \partial_\theta \left(Q_{\tau\phi} + Q_{\rho\phi}\right)$
$i = 9$	$\partial_\phi \left(Q_{\tau\theta} - \frac{1-H}{1+H}Q_{\rho\theta}\right) - \partial_\theta \left(Q_{\tau\phi} - \frac{1-H}{1+H}Q_{\rho\phi}\right)$
$i = 10$	$D_1 \left(s^2 Q_{\theta\theta} - Q_{\phi\phi}\right) - 2sD_2 \left(sQ_{\theta\phi}\right)$

Appendix C. Coupling Matrix for the the Lorenz Gauge

In Equation 5.11 the biggest change from the evolution equations in the Regge-Wheeler-Zerilli gauge is the addition of the coupling matrix $M^{(i)}_{(j)}$. This term is effectively just a nice way of expressing all the terms which do not cleanly fit into the potential or principal part terms. In this appendix I give the coupling terms $M^{(1)}_{(j)}h^{(j)}$ for the tortoise region. I also describe how to convert these into the correct form for the hyperboloidal regions. I use the definitions $\lambda_1^2 \equiv \ell(\ell+1)$ and $\lambda_2^2 \equiv \lambda_1^2(\ell-1)(\ell+2)$ from Appendix A.

$$
\begin{aligned}
M^{(1)}_{(j)}h^{(j)} &= ff'\left(\partial_t + \partial_{r_*}\right)h^{(3)} + \frac{f}{r^3}\left[(r-4M)(2h^{(1)} - 2h^{(4)} - fh^{(3)}) - (r-6M)fh^{(6)}\right] \\
M^{(2)}_{(j)}h^{(j)} &= 2f'\left(\partial_t + \partial_{r_*}\right)h^{(2)} + \frac{f^2}{r^2}\left[2h^{(2)} + 2h^{(5)} - f\left(h^{(3)} + h^{(6)}\right)\right] \\
M^{(3)}_{(j)}h^{(j)} &= \frac{2}{r}\left(\partial_t + \partial_{r_*}\right)h^{(2)} + \frac{f}{r}\left(\partial_t - \partial_{r_*}\right)h^{(3)} + \frac{2f}{r^2}\left(h^{(4)} - h^{(1)} + \frac{r-6M}{r}h^{(3)} - rf'h^{(6)}\right) \\
M^{(4)}_{(j)}h^{(j)} &= f'\left(\partial_t + \partial_{r_*}\right)h^{(5)} - \frac{\lambda_1^2 f}{r^2}\left(2h^{(1)} - fh^{(3)}\right) + \frac{2f}{r^3}(r-7M)h^{(4)} \\
&\quad - \frac{f}{r^3}(r-4M)\left(2h^{(5)} - \lambda_1^2 fh^{(6)} + h^{(7)}\right) \\
M^{(5)}_{(j)}h^{(j)} &= f'\left(\partial_t + \partial_{r_*}\right)h^{(5)} + \frac{2\lambda_1^2 f}{r^2}h^{(2)} + \frac{2f}{r^3}(r-5M)h^{(5)} \\
&\quad - \left(\frac{f}{r}\right)^2\left[\lambda_1^2\left(h^{(3)} + h^{(6)}\right) + 2h^{(4)} - h^{(7)}\right] \\
M^{(6)}_{(j)}h^{(j)} &= \frac{2f}{r^3}(r-4M)\left(h^{(4)} + h^{(6)}\right) - \frac{2f}{r^2}\left(h^{(1)} + h^{2)} - h^{(4)} + h^{(5)}\right) \\
M^{(7)}_{(j)}h^{(j)} &= -\frac{2f}{r^2}\left[\lambda_2^2\left(h^{(4)} - h^{(5)}\right) + h^{(7)}\right] \\
M^{(8)}_{(j)}h^{(j)} &= f'\left(\partial_t + \partial_{r_*}\right)h^{(9)} + \frac{f}{r^3}\left[2(r-7M)h^{(8)} - (r-4M)\left(2h^{(9)} + h^{(10)}\right)\right] \\
M^{(9)}_{(j)}h^{(j)} &= f'\left(\partial_t + \partial_{r_*}\right)h^{(9)} + \frac{2f}{r^3}(r-5M)h^{(9)} - \left(\frac{f}{r}\right)^2\left(2h^{(8)} - h^{(10)}\right) \\
M^{(10)}_{(j)}h^{(j)} &= -\frac{2f}{r^2}\left[\lambda_2^2\left(h^{(8)} - h^{(9)}\right) + h^{(10)}\right]
\end{aligned}
$$

The conversion from these to the outer layer is very simple, except for $i = 3$. For every other component, the $h^{(i)}$ terms are divided by $1 - H^2$, and the $(\partial_t + \partial_{r_*})h^{(i)}$ terms are divided by $1 + H$. The third component differs because the damping term includes $1 - H$. This

equation is

$$M^{(3)}_{(j)}h^{(j)} = \frac{2(1-H)}{r(1+H)}\left(\partial_t + \partial_{r_*}\right)h^{(2)} + \frac{f}{r}(\partial_t - \frac{1-H}{1+H}\partial_{r_*})h^{(3)} + \frac{2f}{r^2}\left(h^{(4)} - h^{(1)} - H\left(h^{(2)} + h^{(5)}\right) + \frac{r-6M}{r}h^{(3)} - \left(rf' - fH\right)h^{(6)}\right)$$

The conversion to the inner layer is more complicated due to the c_i I introduce. Here, I take the choice of $c_1 = 0$ and $c_2 = c_3 = c$. Equations which do not have damping or needed to cancel terms again have simple transformations. For components which are not listed, $h^{(i)}$ terms are divided by $1 - H^2$, the $\partial_t h^{(i)}$ terms are divided by $1 + H$, and $\partial_{r_*}h^{(i)}$ terms are divided by $1 - H$. Due to the complexity of the inner layer, I introduce the notation $H^+ \equiv 1 + H$, $H^- \equiv 1 - H$, and $H^\pm \equiv 1 - H^2$.

$$\begin{aligned}
M^{(2)}_{(j)}h^{(j)} &= \frac{2Hf'}{H^-}(\partial_\tau - \partial_\rho)h^{(1)} + 2f'\left(\partial_\tau + \frac{H^+}{H^-}\partial_\rho\right)h^{(2)} + \frac{2ff'}{H^\pm}\partial_t h^{(3)} \\
&\quad + \frac{f}{r^3H^\pm}[4MH(-h^{(1)} + h^{(2)} + h^{(4)} + h^{(5)}) \\
&\qquad + rf(2h^{(2)} + 2h^{(5)} - f(h^{(3)} + h^{(6)}))] \\
M^{(3)}_{(j)}h^{(j)} &= \frac{2(1-cH)}{r}\left(\partial_\tau + \frac{H^+}{H^-}\partial_\rho\right)h^{(2)} + \frac{fH^+}{H^-}(1-cH)(\partial_\tau - \partial_\rho)h^{(3)} \\
&\quad - \frac{f}{rH^\pm}\left[2f'\left(h^{(3)} + h^{(6)}\right) + \frac{1}{r}\left(2h^{(1)} - fh^{(3)} - 2h^{(4)}\right)\right] \\
&\quad - \frac{fH(1-c-cH)}{r^2H^\pm}\left(2h^{(2)} + 2h^{(5)} - f(h^{(3)} + h^{(6)})\right) \\
M^{(4)}_{(j)}h^{(j)} &= \frac{f'H(1-c-cH)}{H^+}\left(\partial_\tau - \partial_\rho\right)h^{(5)} + f'(1-cH)\left(\partial_\tau + \frac{H^+}{H^-}\partial_\rho\right)h^{(5)} \\
&\quad - \frac{\lambda_1^2 f}{r^2H^\pm}\left(2h^{(1)} - fh^{(3)}\right) - \frac{f}{r^2H^\pm}(rf'(1 + H^+(1-cH)) + 1 - 3f)h^{(4)} \\
&\quad - \frac{f}{r^2H^\pm}(f - rf'H^+(1-cH))(2h^{(5)} - \lambda_1^2 fh^{(6)} + h^{(7)})
\end{aligned}$$

$$
\begin{aligned}
M^{(5)}_{(j)}h^{(j)} &= \frac{f'H}{H^-}(\partial_\tau-\partial_\rho)h^{(4)}+f'(\partial_\tau+\frac{H^+}{H^-}\partial_\rho)h^{(5)}+\frac{2\lambda_1^2 f}{r^2H^\pm}h^{(2)}\\
&\quad+\frac{2f}{r^3H^\pm}(r-5M)h^{(5)}-\frac{ff'H}{rH^\pm}\left(2h^{(4)}-2h^{(5)}+\lambda_1^2h^{(6)}-h^{(7)}\right)\\
&\quad-\frac{f^2}{r^2H^\pm}\left[\lambda_1^2\left(h^{(3)}+h^{(6)}\right)+2h^{(4)}-h^{(7)}\right]\\
M^{(8)}_{(j)}h^{(j)} &= \frac{f'H(1-c-cH)}{H^-}(\partial_\tau-\partial_\rho)h^{(8)}+f'(1-cH)(\partial_\tau+\frac{H^+}{H^-}\partial_\rho)h^{(9)}\\
&\quad+\frac{f}{r^2H^\pm}\left[2(f-\frac{3M}{r})h^{(8)}-f\left(2h^{(9)}+h^{(10)}\right)\right]\\
&\quad-\frac{ff'H^+}{rH^\pm}\left(1-cH\right)\left(2h^{(8)}-2h^{(9)}+h^{(10)}\right)\\
M^{(9)}_{(j)}h^{(j)} &= \frac{f'H}{H^-}(\partial_\tau-\partial_\rho)h^{(8)}+f'(\partial_\tau+\frac{H^+}{H^-}\partial_\rho)h^{(9)}+\frac{2f}{r^3}(r-5M)h^{(9)}\\
&\quad-\frac{f}{r^2H^\pm}(f+rf'H)\left(2h^{(8)}-2h^{(9)}-h^{(10)}\right)-\frac{6Mf}{r^3H^\pm}h^{(9)}
\end{aligned}
$$

Appendix D. Copyright Information

Permission Document for Chapter 2

The contents of Chapter 2 are taken from the paper "The PreSync Project: Synchronization Automation in the Cactus Framework" [2] in the conference journal *PEARC '19: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. The DOI address for this paper is here. This paper was published by the Association for Computing Machinery on the ACM Digital Library. The article preview and associated copyright information is given in Figure D.1.

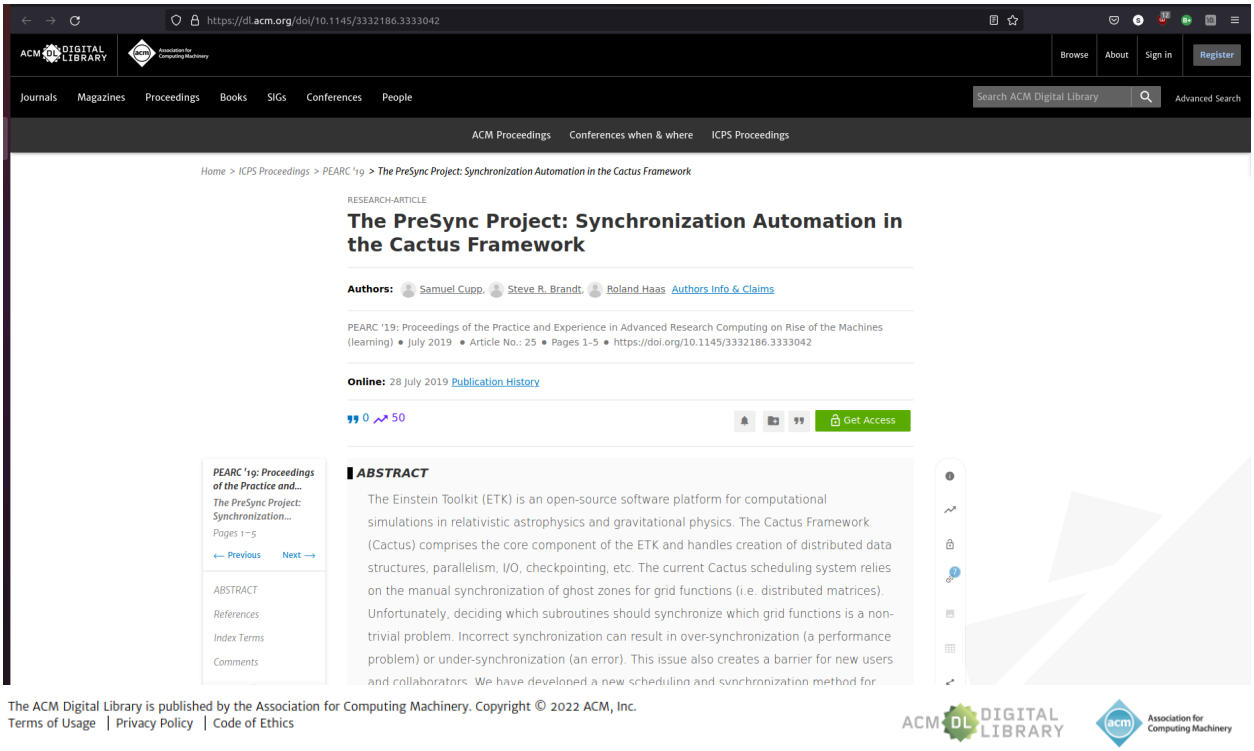

Figure D.1. Snapshot of Publication used in Chapter 2

As per the Author Rights for the Association for Computing Machinery (shown in Figure D.2), the contents of the paper may be used for dissertations.

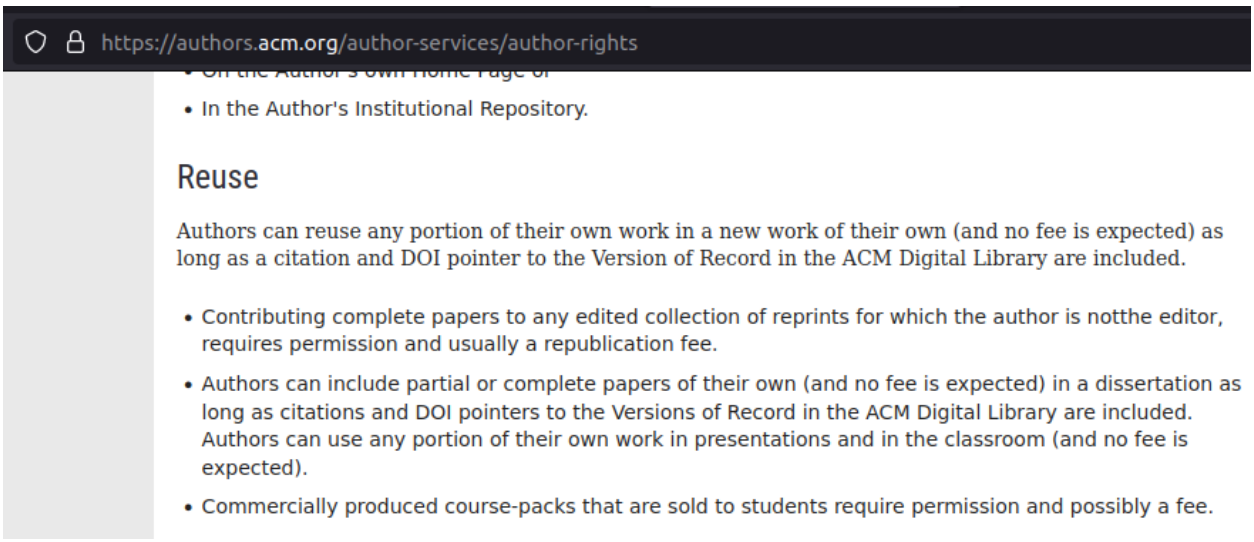

Figure D.2. Snapshot of ACM Author Rights agreement

References

[1] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), "Observation of gravitational waves from a binary black hole merger", Phys. Rev. Lett. **116**, 061102 (2016).

[2] S. Cupp, S. R. Brandt, and R. Haas, "The presync project: synchronization automation in the cactus framework", in Proceedings of the practice and experience in advanced research computing on rise of the machines (learning), PEARC '19 (2019), 25:1–25:5.

[3] S. R. Brandt, G. Bozzola, C.-H. Cheng, P. Diener, A. Dima, W. E. Gabella, M. Gracia-Linares, R. Haas, Y. Zlochower, M. Alcubierre, D. Alic, G. Allen, M. Ansorg, M. Babiuc-Hamilton, L. Baiotti, W. Benger, E. Bentivegna, S. Bernuzzi, T. Bode, B. Brendal, B. Bruegmann, M. Campanelli, F. Cipolletta, G. Corvino, S. Cupp, R. D. Pietri, H. Dimmelmeier, R. Dooley, N. Dorband, M. Elley, Y. E. Khamra, Z. Etienne, J. Faber, T. Font, J. Frieben, B. Giacomazzo, T. Goodale, C. Gundlach, I. Hawke, S. Hawley, I. Hinder, E. A. Huerta, S. Husa, S. Iyer, D. Johnson, A. V. Joshi, W. Kastaun, T. Kellermann, A. Knapp, M. Koppitz, P. Laguna, G. Lanferman, F. Löffler, J. Masso, L. Menger, A. Merzky, J. M. Miller, M. Miller, P. Moesta, P. Montero, B. Mundim, A. Nerozzi, S. C. Noble, C. Ott, R. Paruchuri, D. Pollney, D. Radice, T. Radke, C. Reisswig, L. Rezzolla, D. Rideout, M. Ripeanu, L. Sala, J. A. Schewtschenko, E. Schnetter, B. Schutz, E. Seidel, E. Seidel, J. Shalf, K. Sible, U. Sperhake, N. Stergioulas, W.-M. Suen, B. Szilagyi, R. Takahashi, M. Thomas, J. Thornburg, M. Tobias, A. Tonita, P. Walker, M.-B. Wan, B. Wardell, L. Werneck, H. Witek, M. Zilhão, and B. Zink, *The einstein toolkit*, version The "Katherine Johnson" release, ET_2021_11, To find out more, visit http://einsteintoolkit.org, Dec. 2021.

[4] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, "The cactus framework and toolkit: design and applications", in *High performance computing for computational science — vecpar 2002*, Vol. 2565, edited by J. M. Palma, A. A. Sousa, J. Dongarra, and V. Hernández, Lecture Notes in Computer Science (Springer-Verlag, Berlin, 2002), pp. 197–227.

[5] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations", Journal of Computational Physics **53**, 484–512 (1984).

[6] J. Bordner and M. L. Norman, "Enzo-p/cello: scalable adaptive mesh refinement for astrophysics and cosmology", in Proceedings of the extreme scaling workshop (University of Illinois at Urbana-Champaign, 2012), p. 4.

[7] G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, *et al.*, "Enzo: an adaptive mesh refinement code for astrophysics", The Astrophysical Journal Supplement Series **211**, 19 (2014).

[8] L. V. Kale and S. Krishnan, *Charm++: a portable concurrent object oriented system based on c++*, Vol. 28 (Citeseer, 1993).

[9] D. C. Marcello, K. Kadam, G. C. Clayton, J. Frank, H. Kaiser, and P. M. Motl, "Introducing octo-tiger/hpx: simulating interacting binaries with adaptive mesh refinement and the fast multipole method", Proceedings of Science, 13–17 (2016).

[10] H. Kaiser, T. Heller, B. Adelstein-Lelbach, A. Serio, and D. Fey, "Hpx: a task based programming model in a global address space", in Proceedings of the 8th international conference on partitioned global address space programming models (ACM, 2014), p. 6.

[11] Y. Mino, M. Sasaki, and T. Tanaka, "Gravitational radiation reaction to a particle motion", Phys. Rev. D **55**, 3457–3476 (1997).

[12] T. C. Quinn and R. M. Wald, "Axiomatic approach to electromagnetic and gravitational radiation reaction of particles in curved spacetime", Phys. Rev. D **56**, 3381–3394 (1997).

[13] S. Gralla and R. Wald, "A rigorous derivation of gravitational self-force", Class. Quantum Grav. **25**, 205009 (2008).

[14] E. Pound, "Retarded coordinates based at a world line and the motion of a small black hole in an external universe", Phys. Rev. D **69**, 084007 (2004).

[15] A. Pound, "Self-consistent gravitational self-force", Phys. Rev. D **81**, 024023 (2010).

[16] T. C. Quinn and R. M. Wald, "Axiomatic approach to electromagnetic and gravitational radiation reaction of particles in curved spacetime", Phys. Rev. D **56**, 3381–3394 (1997).

[17] E. Poisson, A. Pound, and I. Vega, "The motion of point particles in curved spacetime", Living Reviews in Relativity **14**, 7 (2011).

[18] S. Detweiler and B. F. Whiting, "Self-force via a green's function decomposition", Phys. Rev. D **67**, 024025 (2003).

[19] S. Detweiler, E. Messaritaki, and B. F. Whiting, "Self-force of a scalar field for circular orbits about a schwarzschild black hole", Phys. Rev. D **67**, 104016 (2003).

[20] R. Haas and E. Poisson, "Mode-sum regularization of the scalar self-force: formulation in terms of a tetrad decomposition of the singular field", Phys. Rev. D **74**, 044009 (2006).

[21] L. Barack and A. Ori, "Gravitational self-force on a particle orbiting a kerr black hole", Phys. Rev. Lett. **90**, 111101 (2003).

[22] M. Casals, S. Dolan, A. C. Ottewill, and B. Wardell, "Self-force and green function in schwarzschild spacetime via quasinormal modes and branch cut", Phys. Rev. D **88**, 044022 (2013).

[23] N. Warburton and B. Wardell, "Applying the effective-source approach to frequency-domain self-force calculations", Phys. Rev. D **89**, 044046 (2014).

[24] D. V. Gal'tsov, "Radiation reaction in the kerr gravitational field", Journal of Physics A: Mathematical and General **15**, 3737–3749 (1982).

[25] Y. Mino, "Perturbative approach to an orbital evolution around a supermassive black hole", Phys. Rev. D **67**, 084027 (2003).

[26] N. Sago, T. Tanaka, W. Hikida, K. Ganz, and H. Nakano, "Adiabatic Evolution of Orbital Parameters in Kerr Spacetime", Progress of Theoretical Physics **115**, 873–907 (2006).

[27] R. Geroch and P. S. Jang, "Motion of a body in general relativity", Journal of Mathematical Physics **16**, 65–67 (1975).

[28] N. Sago, T. Tanaka, W. Hikida, and H. Nakano, "Adiabatic Radiation Reaction to Orbits in Kerr Spacetime", Progress of Theoretical Physics **114**, 509–514 (2005).

[29] É. É. Flanagan and T. Hinderer, "Transient resonances in the inspirals of point particles into black holes", Phys. Rev. Lett. **109**, 071102 (2012).

[30] A. Pound and E. Poisson, "Multiscale analysis of the electromagnetic self-force in a weak gravitational field", Phys. Rev. D **77**, 044012 (2008).

[31] A. Pound and E. Poisson, "Osculating orbits in schwarzschild spacetime, with an application to extreme mass-ratio inspirals", Phys. Rev. D **77**, 044013 (2008).

[32] A. Pound, E. Poisson, and B. G. Nickel, "Limitations of the adiabatic approximation to the gravitational self-force", Phys. Rev. D **72**, 124001 (2005).

[33] I. Vega, B. Wardell, P. Diener, S. Cupp, and R. Haas, "Scalar self-force for eccentric orbits around a schwarzschild black hole", Phys. Rev. D **88**, 084021 (2013).

[34] L. Barack and D. A. Golbourn, "Scalar-field perturbations from a particle orbiting a black hole using numerical evolution in $2 + 1$ dimensions", Phys. Rev. D **76**, 044020 (2007).

[35] L. Barack, D. A. Golbourn, and N. Sago, "m-mode regularization scheme for the self-force in kerr spacetime", Phys. Rev. D **76**, 124036 (2007).

[36] S. R. Dolan and L. Barack, "Self-force via m-mode regularization and $2+1$D evolution: foundations and a scalar-field implementation on schwarzschild spacetime", Phys. Rev. D **83**, 024019 (2011).

[37] I. Vega and S. Detweiler, "Regularization of fields for self-force problems in curved spacetime: foundations and a time-domain application", Phys. Rev. D **77**, 084008 (2008).

[38] I. Vega, P. Diener, W. Tichy, and S. Detweiler, "Self-force with $(3+1)$ codes: a primer for numerical relativists", Phys. Rev. D **80**, 084021 (2009).

[39] I. Vega, "The dynamics of point particles around black holes", PhD thesis (University of Florida, Gainsville, July 2009).

[40] S. Bernuzzi, A. Nagar, and A. Zenginoğlu, "Binary black hole coalescence in the large-mass-ratio limit: the hyperboloidal layer method and waveforms at null infinity", Phys. Rev. D **84**, 084026 (2011).

[41] C. E. Grosch and S. A. Orszag, "Numerical solution of problems in unbounded regions: coordinate transforms", Journal of Computational Physics **25**, 273–295 (1977).

[42] A. Nagar, J. A. Font, O. Zanotti, and R. D. Pietri, "Gravitational waves from oscillating accretion tori: comparison between different approaches", Phys. Rev. D **72**, 024007 (2005).

[43] J. S. Hesthaven and T. Warburton, *Nodal discontinuous galerkin methods* (Springer-Verlag, New York, 2008).

[44] R. Courant, E. Isaacson, and M. Rees, "On the solution of nonlinear hyperbolic differential equations by finite differences", Communications on Pure and Applied Mathematics **5**, 243–255 (1952).

[45] J. S. Hesthaven, "From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex", SIAM Journal on Numerical Analysis **35**, 655–676 (1998).

[46] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb, *Spectral methods for time-dependent problems* (Cambridge University Press, Jan. 2007).

[47] K. Martel and E. Poisson, "Gravitational perturbations of the schwarzschild spacetime: a practical covariant and gauge-invariant formalism", Phys. Rev. D **71**, 104003 (2005).

[48] T. Regge and J. A. Wheeler, "Stability of a schwarzschild singularity", Phys. Rev. **108**, 1063–1069 (1957).

[49] F. J. Zerilli, "Gravitational field of a particle falling in a schwarzschild geometry analyzed in tensor harmonics", Phys. Rev. D **2**, 2141–2160 (1970).

[50] V. Moncrief, "Gravitational perturbations of spherically symmetric systems. i. the exterior problem", Annals of Physics **88**, 323–342 (1974).

[51] F. K. Martel, "Particles and black holes: Time-domain integration of the equations of black-hole perturbation theory", PhD thesis (University of Guelph, Canada, Jan. 2004).

[52] C. T. Cunningham, R. H. Price, and V. Moncrief, "Radiation from collapsing relativistic stars. I - Linearized odd-parity radiation", Astrophys. J. **224**, 643 (1978).

[53] S. Jhingan and T. Tanaka, "Improvement on the metric reconstruction scheme in Regge-Wheeler-Zerilli formalism", Phys. Rev. D **67**, 104018 (2003).

[54] K. S. Thorne, "Multipole expansions of gravitational radiation", Rev. Mod. Phys. **52**, 299–339 (1980).

[55] E. Poisson, "Absorption of mass and angular momentum by a black hole: time-domain formalisms for gravitational perturbations, and the small-hole or slow-motion approximation", Phys. Rev. D **70**, 084044 (2004).

[56] L. Barack and C. O. Lousto, "Perturbations of schwarzschild black holes in the lorenz gauge: formulation and numerical implementation", Phys. Rev. D **72**, 104026 (2005).

[57] L. Barack and N. Sago, "Gravitational self-force on a particle in circular orbit around a schwarzschild black hole", Phys. Rev. D **75**, 064021 (2007).

[58] C. Gundlach, G. Calabrese, I. Hinder, and J. M. Martín-García, "Constraint damping in the z4 formulation and harmonic gauge", Classical and Quantum Gravity **22**, 3767–3773 (2005).

[59] E. Berti, V. Cardoso, and A. O. Starinets, "Quasinormal modes of black holes and black branes", Classical and Quantum Gravity **26**, 163001 (2009).

[60] *Black hole perturbation toolkit*, https://bhptoolkit.org.

[61] L. C. Stein, "qnm: A Python package for calculating Kerr quasinormal modes, separation constants, and spherical-spheroidal mixing coefficients", J. Open Source Softw. **4**, 1683 (2019).

Vita

Samuel Cupp is a native of Maryville, Tennessee. As a high schooler, he attended the Tennessee Governor's School of Computational Physics at Austin Peay State University, which led to his eventual choice of school and major. He earned a Bachelor's degree from Austin Peay State University with a major in physics and a minor in mathematics. As with his undergraduate institution, his participation in two summers of the Research Experience for Undergraduates program at Louisiana State University led to his later choice of graduate school. He decided to pursue a doctorate focusing on computational gravitational research at Louisiana State University. Upon completion of his doctorate degree he will continue working on open-source computational tools for the simulation of astrophysical systems.