ABSTRACT

LOWE, LISA L. Topics in Numerical Relativity: Solving the Initial Value Problem Using Adaptive Mesh Refinement, Examining Evolution Stability Using Spectral Methods, and Finding Apparent Horizons using a Mean Curvature–Level Set Method. (Under the direction of Associate Professor J. David Brown).

Various topics in numerical relativity will be discussed, including solving the initial value problem using adaptive mesh refinement, examining evolution stability using spectral methods, and finding apparent horizons using a mean curvature–level set method. We use the ADM 3+1 formalism, which separates the Einstein equations into four constraint equations and twelve evolution equations. The four constraint equations are then transformed via a York–Lichnerowicz transverse traceless decomposition. We give a brief discussion on variations of this decomposition. We discuss the implementation of a parallel multigrid solver with adaptive mesh refinement for solving the initial value constraint equations. Using our multigrid solver, we solve for a new class of initial data: distorted black holes using the puncture method. The ADM evolution equations have known instabilities. We explore the instabilities inherent in the evolution equations with an evolution code implemented using spectral methods with a Runge–Kutta integrator via the method of lines. An overview of spectral methods is given. We compare the results of evolving the full ADM evolution equations with stability predictions from integrating the constraint equations. Evolution instabilities can be contained by adding multiples of the constraints to the evolution equations. We show how to implicitly dictate the behavior of constraint violation during an evolution by manipulating additional constraint terms. After solving for initial data or during an evolution, it is often necessary to locate the position of the apparent horizon. We rewrite the apparent horizon equation as a surface evolving along its normal vector according to the speed of the apparent horizon equation. Instead of evolving a two dimensional surface and assuming a star shaped topology, we evolve a three dimensional surface via the level set method. This allows us to find multiple apparent horizons of any topology in generic spacetimes, either analytic or numerically generated.

Topics in Numerical Relativity: Solving the Initial Value Problem Using Adaptive Mesh Refinement, Examining Evolution Stability Using Spectral Methods, and Finding Apparent Horizons using a Mean Curvature–Level Set Method

by
Lisa L. Lowe

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Physics

Raleigh, North Carolina

2008

APPROVED BY:

Dr. John M. Blondin	Dr. Gail McLaughlin
Dr. J. David Brown Chair of Advisory Committee	Dr. Stephen P. Reynolds

DEDICATION

To Ivan.

BIOGRAPHY

Lisa Lowe attended Belle Vernon Area High School in southwestern Pennsylvania, where she studied physics under Mr. Thomas C. Thompson Jr. She obtained her undergraduate degree in physics from Drexel University in Philadelphia, Pennsylvania. At Drexel, she worked with Professor Joan M. Centrella on hydrodynamic simulations of instabilities in rapidly rotating polytrope stars [1]. Lisa completed a co–op at the Princeton Plasma Physics Laboratory in Plainsboro, New Jersey where she implemented a parallelization of a guiding center drift orbit integrator [2] and developed a field line tracing routine for a resistive magnetohydrodynamics code with finite elements on unstructured meshes [3]. She attended graduate school at the North Carolina State University in Raleigh where she worked as the head teaching assistant for Astronomy Labs and for a semester was the Interim Manager of the Physics Tutorial Center. At NC State, she did research in the field of numerical relativity with Professor J. David Brown. Her research included implementing a multigrid solver with adaptive mesh refinement [4], solving initial data for distorted black holes using the puncture method [5], investigating the stability of the Einstein equations with spectral methods [6], and finding apparent horizons with a mean curvature flow level set method. She is married to Ivan and has two daughters, Lenore and Adrijana.

ACKNOWLEDGMENTS

I would like to thank my advisor David Brown for sharing his expert knowledge with me; I only wish I could have absorbed more of it during my studies. He has made several attempts to push me away from blind numerical experimentation towards mathematical rigor and careful thought. Although I am light-years away from achieving his level of excellence, I thank him for providing the standard that I will work toward. On a more personal level, I am grateful to have had an advisor with an open door policy and the ability to patiently explain (sometimes repeatedly) complicated ideas and concepts. He and his wife Stefanie have also been great friends to me, and they have been an endless reservoir of support for me and my family since the day we moved into the area.

I am grateful for all the sources of funding I have received during my graduate studies. All course work was completed while being supported by a Teaching Assistantship from the Physics Department at NC State. I give special thanks to Robert Egler for supporting me in my position as Astronomy teaching assistant and head teaching assistant. A GAANN fellowship enabled me to take a break from teaching to study for the qualifying exam while also having time to work on coding. Work developing the multigrid solver, the spectral methods solver, and the apparent horizon finder was supported by Research Assistant funding from Dr. J. David Brown. The writing of this dissertation was supported by an American Fellowship from the American Association of University Women.

I would like to express my sincere gratitude to all of my previous advisors and mentors, for without them I would not have had the confidence and experience necessary to have achieved what I have. In particular, Joan Centrella not only paved my way into computational physics and relativity, but took me by the hand down the path. As a freshman undergraduate, I had a work-study job doing filing and secretarial work for the Physics department. She came by and handed me a Fortran manual and asked me if I wanted a job doing real physics. Other than offering me invaluable experience and connections, I still often recall her words of wisdom and encouragement during frustrating encounters. Xianzhu Tang was another amazing mentor that I was fortunate to have. He not only taught me some interesting computational physics but also gave sage advices for confidently giving presentations in an intense environment. Finally, I would like to thank Tom Thompson for being an inspirational teacher. Most students either love or despise physics based on the performance of their first physics instructor. Here is no exception; if Tom had made a different career choice, I might have as well.

Most of all, I am grateful to my family for making everything worthwhile. My mother-in-law Nada took care of my children and household during the final crunch to finish. My daughters gave me much needed comic relief, Lena by scribbling over my notes and yelling "No Dissertation!", and Jana by flashing her adorable toothless grin. My husband Ivan not only provided his love and support, but also read and offered insightful suggestions on each of the several dozen drafts of this text.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Einstein published the field equations of general relativity in 1915. This caused a shift in gravity theory from a Newtonian paradigm where objects move due to the mysterious forces acting between them into a much simpler paradigm: objects go straight. Matter causes space to curve, and all matter travels on straight lines in this curved space. The first exact solution to the field equations was published shortly after the equations themselves. In 1916, Karl Schwarzschild found a solution for a spherically symmetric stationary object. The simple idea of objects going straight becomes rather complicated due to the 'in curved space' caveat, so the work of finding exact solutions is difficult. Simplifying assumptions and special symmetries, such as the spherical symmetry of a Schwarzschild metric, are used to find such solutions. Beyond exact solutions, analytic approximation techniques can be used to further examine the field equations. These analytic approximations require considerable simplifications as well. A computer is necessary to solve the Einstein equations for anything significantly complex or realistic. However, despite the advent of powerful supercomputers and state-of-the-art numerical techniques, the two-body problem in full general relativity (two gravitationally interacting objects), which is such a trivial scenario in the Newtonian realm, remained elusive for many years. It was nearly a century after the Einstein field equations were first published that the first successful computer simulations solving the two-body problem were achieved.

In the quest to solve the two-body problem, a new branch of science was created—

numerical relativity. Numerical relativity is the art of solving the Einstein equations using a computer. The first 'numerical relativists' were Hahn and Lindquist in 1964 [7]. This first attempt at modeling a two black hole collision was not very successful, and not surprisingly so, since everything about the problem was in its infancy. Computers were primitive, as was the science of numerical modeling in general. Even the analytic side of the problem was not fully developed. At that time, the phrase 'black hole' did not yet exist. The term 'black hole' was popularized by John Wheeler in 1967. Instead, the term 'geometrodynamics' was used in this first numerical relativity publication. Although unsuccessful in modeling a black hole merger, Hahn and Lindquist successfully inspired other scientists to attack the two-body problem. In 1970, Dewitt, Smarr, and Eppley published the results of a simulation that followed two black holes as they merged into one [8].

As the fledgling science of numerical relativity struggled to take wing, the theoretical implications of general relativity were already becoming well developed and understood [9]. The theory had predicted the perihelion shift of Mercury and the bending of light during an eclipse. Several experiments also confirmed the prediction of a gravitational redshift. Furthermore, it could be shown through a rather simple calculation that moving objects should cause gravitational waves [10]. Gravitational waves are ripples in the very fabric of spacetime. There was not a great deal of credence given to the idea of gravitational waves until the unexpected discovery of an oddly behaving pulsar by graduate student Russell Hulse in 1974 [11]. The deviation from the expected intervals of radio signals from the Hulse–Taylor Binary Pulsar could be explained by energy being dissipated as gravitational waves. This discovery implied that gravitational waves did indeed exist and thus could be detected and studied. Another branch of science was then created—gravitational wave science. The quest to detect gravitational waves had begun, resulting in a collaboration by Caltech and MIT in 1983 to build a gravitational wave detector LIGO (Laser Interferometer Gravitational Wave Observatory) [12, 13]. The project received funding by National Science Foundation (NSF) in 1992, with the groundbreaking in 1996. Calibrations and improvements are still being made, with the possibility of a space-based interferometer, LISA (Laser Interferometer Space Antennae), in the works. As of the time of this writing, NASA has set an earliest launch date estimate for LISA as 2018 [14].

In the 1990's, anticipation for LIGO inspired researchers to pursue with renewed vigor the two-body problem of general relativity. The Black Hole Binary Grand Challenge Alliance was formed and funded in 1993. The goal of the alliance was to build up numerical algorithms and infrastructure to successfully produce a three-dimensional model of the inspiral, merger, and resulting gravitational waves of two black holes[1]. During the Grand Challenge era, great volumes of knowledge were acquired. Many computer programs were written and rigorous testbeds were developed for verifying the accuracy and stability of algorithms [16]. Alternate mathematical representations of the equations were formulated to enhance stability [17]. Techniques were developed to handle the inherent singularities of black hole spacetimes(e.g., excision [18], punctures [19], gauge choices [20]). On the computational front, faster, more efficient algorithms and parallel adaptive grid infrastructure were being developed while the availability of supercomputers was becoming more widespread. By 1998, stable long running evolutions of single black holes were possible [21], but similar success with binary black holes still seemed a long way off. At the end of the 1990's, many researchers began to study in detail the stability issues inherent in the formulation of the Einstein equations [22]. A great deal of work was done to understand issues such as the mathematical classification of the equations (such as hyperbolicity) and how to formulate appropriate boundary conditions. Meanwhile, the library of numerical algorithms continued to grow. Adaptive mesh refinement, parallel processing, and better computational approximations of equations (higher order finite differencing, spectral methods) were becoming well developed. This new wealth of knowledge led to two simultaneous breakthroughs, both for the first time succeeding in solving the elusive two-body problem of general relativity. The first came from the understanding that the standard formulations of relativity were, in general, ill-suited to solve the problem at hand. Most work on the stability of formulations entailed modifications of the usual (ADM) equations, but in 2004, work using a different formulation of the Einstein equations (known as 'generalized harmonic coordinates') led to the first sim-

[1]NCSA and the University of Illinois has developed an easy to follow website with more details on the history of numerical relativity as well as movies from various simulations [15].

ulation of a black hole binary inspiral and merger [23, 24]. The second breakthrough surfaced as a culmination of relativity work over the years. In 2005, two research groups were able to take a binary system through inspiral and merger via a 'moving puncture method'. The moving puncture method is successful by means of a modified form of the equations (BSSN), specialized gauge choices, and advanced computational techniques. The computational technique of adaptive mesh refinement was used to allow the resolution needed to evolve highly singular objects (punctures) [25, 26, 27] while providing the ability to place boundary conditions far from the source. Since these successful simulations were published, researchers have been developing their existing codes and writing new ones to run their own successful simulations using either the moving puncture or generalized harmonic method[2].

This dissertation touches on a representative sampling of the issues in numerical relativity. Chapter 2 describes some of the necessary theoretical background including the Einstein equations and the 3+1 decomposition of the Einstein equations known as the ADM decomposition. Chapter 3 explains the initial value problem of general relativity. To evolve any physical system, one must start with valid initial data. The initial data must obey the laws of physics; in our case, it must satisfy constraint equations. Chapter 4 will describe our initial value solver using multigrid and adaptive mesh refinement called AMRMG (Adaptive Mesh Refinement MultiGrid). The initial data for one of the first successful binary black hole simulations was provided by AMRMG. Chapter 5 details a new class of initial data made feasible by AMRMG: distorted black holes using the puncture method. Chapter 6 describes the evolution equations as they written in the ADM formalism. It explains the inherent stabilities encountered with this form, and it illustrates exponentially growing instabilities with some numerical examples. This chapter also contains an introduction to controlling constraint violations during an evolution. We use spectral methods to implement a constrained evolution. Spectral methods are well established computational tools, but they are somewhat novel to the numerical relativity community. Chapter 7 explains spectral methods as they are most commonly used in relativity and describes the

[2]For a more detailed synopsis of numerical relativity successes, see various 'Status of Relativity' papers [28, 29, 30, 31, 30, 32].

successes and problems we have encountered in implementing an evolution code using pseudospectral collocation. Chapter 8 is our publication explaining how any system, such as the Einstein evolution equations, might be altered to control the growth of constraint violating modes (i.e., to make the system numerically well behaved). After solving for the configuration of a certain spacetime, it is important to understand its physical characteristics. In particular, it is useful to be able to search for horizons in a spacetime. Chapter 9 will describe an apparent horizon finder using a method that evolves a level set according to its mean curvature or expansion flow. Chapter 10 gives a summary of the three major code projects described in this dissertation, and it outlines several projects for prospective graduate or undergraduate students.

Chapter 2

Equations

There is a dilemma in formulating an introductory chapter on numerical relativity. The likely prospective audience for such a text would be either a graduate student beginning to do research or someone already working in the field. To a veteran of the field, a lengthy explanation of the material is trivial and unnecessary. In light of this, the spirit of the following chapter is to present the material in a way that it is conceptually understandable to an incoming graduate student[1]. Accurate formulas, common notation, and useful references for further study are provided. The aim will be to allow a beginner to jump in and work without getting immediately scared off by the imposing looking and unfamiliar math. A veteran can scan through for the notation conventions used.

2.1 Einstein equations

A relativity dissertation is most appropriately begun with the Einstein equations, thus, to begin we have [2]

$$G_{\mu\nu} = 8\pi T_{\mu\nu}. \tag{2.1}$$

$G_{\mu\nu}$ is the Einstein tensor. It represents curvature. $T_{\mu\nu}$ is the stress energy tensor.

[1] In a similar vein, Miller [33] provides a brief explanation of numerical relativity for the beginner.

[2] Equation (2.1) and subsequent equations will be in geometrized units where the gravitational constant and the speed of light are $G = c = 1$.

It represents matter. This relation is most eloquently explained by John Wheeler: "Spacetime tells mass how to move, and mass tells spacetime how to curve." Space is described by the metric. A metric tensor will be denoted $g_{\mu\nu}$, and a metric will be denoted[3] by $ds^2 \equiv g_{\mu\nu} dx^\mu dx^\nu$. The metric gives the distance between two points. The most common example is a two-dimensional flat metric $ds^2 = dx^2 + dy^2$. This is just the familiar Pythagorean theorem. Here are two examples.
A cylindrical metric:

$$g_{ij} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$ds^2 = d^2\rho + \rho^2 d^2\phi + dz^2$$

The Minkowski metric:

$$g_{\mu\nu} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$ds^2 = -dt^2 + dx^2 + dy^2 + dz^2$$

The cylindrical metric is purely spatial—there is no time information. For spatial metrics, we will use Latin indices running from 1 to 3 with metric signature (+,+,+). The Minkowski Metric is a 'spacetime' metric—it contains information about both space and time. For spacetime metrics, we use Greek indices running from 0 to 3. A spacetime metric has metric signature (-,+,+,+). A metric with such signature is called a 'pseudo-Riemannian metric'.

General relativity is a description of gravity as geometry. Mass creates curvature. Curvature is computed from derivatives of the metric. In differential geometry, the geometry of curved space, a Riemann tensor gives us a measure of curvature. If the

[3]Here we use the Einstein summation notation: sum over repeated indices unless stated otherwise.

Riemann tensor, $R^{\mu}{}_{\nu\alpha\beta}$, equals zero then the curvature is zero, i.e., flat. The Riemann tensor consists of the metric and its derivatives as follows:

$$R^{\mu}{}_{\nu\alpha\beta} \equiv \partial_{\alpha}\Gamma^{\mu}_{\nu\beta} - \partial_{\beta}\Gamma^{\mu}_{\nu\alpha} + \Gamma^{\mu}_{\sigma\alpha}\Gamma^{\sigma}_{\mu\beta} - \Gamma^{\mu}_{\sigma\beta}\Gamma^{\sigma}_{\mu\alpha} \tag{2.4}$$

where the Christoffel symbols, $\Gamma^{\mu}_{\alpha\beta}$, are defined as

$$\Gamma^{\mu}_{\alpha\beta} = \frac{1}{2}g^{\mu\sigma}(\partial_{\alpha}g_{\beta\sigma} + \partial_{\beta}g_{\alpha\sigma} - \partial_{\sigma}g_{\alpha\beta}). \tag{2.5}$$

The Riemann tensor can be contracted to form the Ricci tensor

$$R_{\mu\nu} = R^{\sigma}{}_{\mu\sigma\nu}, \tag{2.6}$$

which can then be contracted to form the Ricci scalar, or scalar curvature

$$R = g^{\mu\nu}R_{\mu\nu}. \tag{2.7}$$

Now we can express the Einstein tensor in terms of the Riemann tensor

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R. \tag{2.8}$$

Compare the field equations of relativity (equation [2.1]) with the familiar Newtonian equation for gravitational potential.

$$G_{\mu\nu} = 8\pi T_{\mu\nu} \tag{2.9}$$

$$\nabla^2\psi = 4\pi\rho \tag{2.10}$$

The Newtonian theory gives the Poisson equation, a differential operator acting on the gravitational potential depending on a matter source term. Recalling that $G_{\mu\nu}$ is made up of the metric and its derivatives, one can see that the Einstein equations are also differential equations. The matter source term is $T_{\mu\nu}$ and the left hand side is a differential operator acting on the metric. In principle, we simply describe the source term and solve the equations for the metric $g_{\mu\nu}$. In practice, however, solving this set of equations is anything but simple. As intimated in the Introduction, solving the Einstein equations is quite a complicated task, even in the absence of source terms (vacuum relativity). In vacuum relativity, the stress energy tensor is zero, allowing spacetimes with black holes and gravitational waves. In this dissertation, only vacuum relativity will be discussed. The equation of interest is now

$$G_{\mu\nu} = 0. \tag{2.11}$$

Unlike its source free Newtonian counterpart (the Laplace equation), there is no straightforward prescription for solving equation (2.11). Equation (2.11) is a set of ten nonlinear coupled second order partial differential equations.

2.2 ADM—The 3+1 Decomposition

It is difficult to imagine where one would start when faced with the set of equations that is equation (2.11). What one likely does is make an attempt to translate the unfamiliar into something more routine. The Einstein tensor gives information for a four-dimensional spacetime. We are more likely to imagine ourselves in a particular place at one time, then visualize that time is moving forward. The ADM (Arnowitt, Deser, Misner) decomposition [34], also known as the 3+1 formalism, is a way to create three spatial dimensions going forward in time. Using ADM allows us to do computations in this familiar way[4]. Technically, the ADM decomposition separates spacetime into a set (or foliation) of spacelike hypersurfaces evolving in time[5].

Doing this separation gives what is known as a Cauchy problem [36]. 'Cauchy problem' basically means 'initial value problem'. Consider the Newtonian 'projectile' initial value problem. If the initial position and momentum of an object are known, the trajectory can be predicted for all time. In the Hamiltonian formulation of classical mechanics, we talk about 'q's and p's' or 'position and momentum' as conjugate variables. In 3+1 relativity, the 'q' is the metric of the spacelike hypersurface g_{ij}[6]. What is the 'p'? We need something that acts like a momentum. The metric

[4]Although the ADM formalism turns the Einstein equations into a 'computer ready' type system of equations, this was not its original purpose. The ADM decomposition came about as an attempt to quantize gravity by defining a general relativistic Hamiltonian.

[5]Reference [35] gives a step-by-step description on how to project the spacetime quantities of the Einstein equations into the three-dimensional hypersurfaces.

[6]In the following, recall that spacetime quantities will have Greek indices and purely spatial quantities will have Latin indices. If operators such as a derivative or Riemann tensor have Latin indices, that means they will be associated with the spacelike metric g_{ij}.

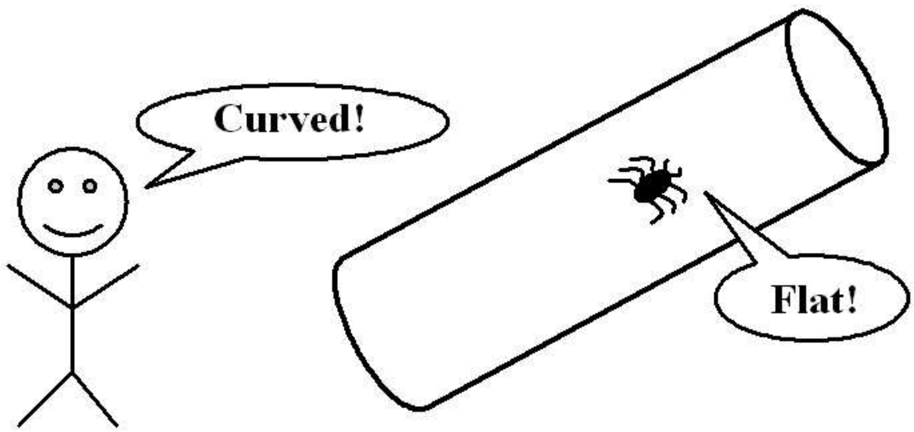

Figure 2.1: Extrinsic vs. Intrinsic curvature.

g_{ij} represents curvature, so what would represent the change in curvature? A brief discussion of curvature will answer this question.

Consider the cylinder in figure 2.1. An ant crawling on the cylinder sees no curvature. In the absence of curvature, two initially parallel lines will never intersect, and the ant will verify that this is true on the cylinder. The type of curvature an ant sees is called 'intrinsic curvature'. It is intrinsic to the manifold. An observer outside of the manifold will insist that the cylinder is curved. The curvature comes about from how the two-dimensional cylinder is embedded in the three-dimensional space of the observer. This is known as 'extrinsic curvature'. An observer can measure how the curvature of the cylinder changes with respect to the surrounding area by placing an arrow normal (or orthogonal) to the surface of the cylinder and moving it around, observing how fast the arrow changes directions. Now replace the cylinder with a three-dimensional spacelike hypersurface. The intrinsic curvature of our hypersurface is g_{ij}, and the extrinsic curvature (our momentum-like variable) will be denoted by K_{ij}. K_{ij} is defined as

$$K_{ij} = -\frac{1}{2}\mathcal{L}_n(g_{ij}) \tag{2.12}$$

where $\mathcal{L}_n(g_{ij})$ is the Lie derivative of g_{ij} along n^μ, where n^μ is the timelike unit normal to the hypersurface.

Imagine we now have a set of spacelike hypersurfaces containing the dynamical variables g_{ij} and K_{ij}. Each hypersurface is defined at an instant of constant time. The dynamical variables evolve in time from one hypersurface to another. The next step is to define the trajectory needed to take a point x^i at time t forward in time to $t + \delta t$. To do this we break up the time vector into components normal to the hypersurface n^μ and tangent to the hypersurface β^μ

$$t^\mu = \alpha n^\mu + \beta^\mu. \tag{2.13}$$

Figure 2.2 shows a diagram depicting this. The lapse α gives the information about the 'time lapsed' from one surface to the next, while the shift vector β^i tells us how the coordinates shift[7]. These gauge conditions are also referred to as 'slicing conditions'. The spacetime is 'sliced' into hypersurfaces, but there is not a unique way of doing this. The easiest thing to imagine is having a block and making horizontal slices straight across. Each point would go directly forward in time ($\alpha = 1$) and there would be no shift in coordinates ($\beta^i = 0$). Although this is the simplest case, it is usually necessary to make a more clever choice for the lapse and shift to avoid instabilities and singularities.

To summarize, in order to create an initial value formulation of relativity, the four-dimensional spacetime equations of relativity are decomposed via the 3+1 split of the ADM formalism. The split is done by defining a foliation of spacelike hypersurfaces and projecting the Einstein equations onto that foliation. In doing so, we obtain a position-like quantity, g_{ij}, and a momentum-like quantity, K_{ij}. The separation results in constraint equations and evolution equations for the dynamical variables g_{ij} and K_{ij}. One must solve the constraint equations to provide valid physical initial data on the initial spacelike hypersurface (t=0). After valid initial data is specified, the evolution equations are used to evolve the data onto future hypersurfaces. To perform an evolution, one must choose freely specifiable gauge conditions, α and β^i.

The ADM metric is expressed as

$$ds^2 \equiv \gamma_{\mu\nu} dx^\mu dx^\nu = (\beta_a \beta^a - \alpha^2) dt^2 + 2\beta_a dx^a dt + g_{ab} dx^a dx^b. \tag{2.14}$$

[7] In some literature, the lapse and shift are denoted N and N^i respectively.

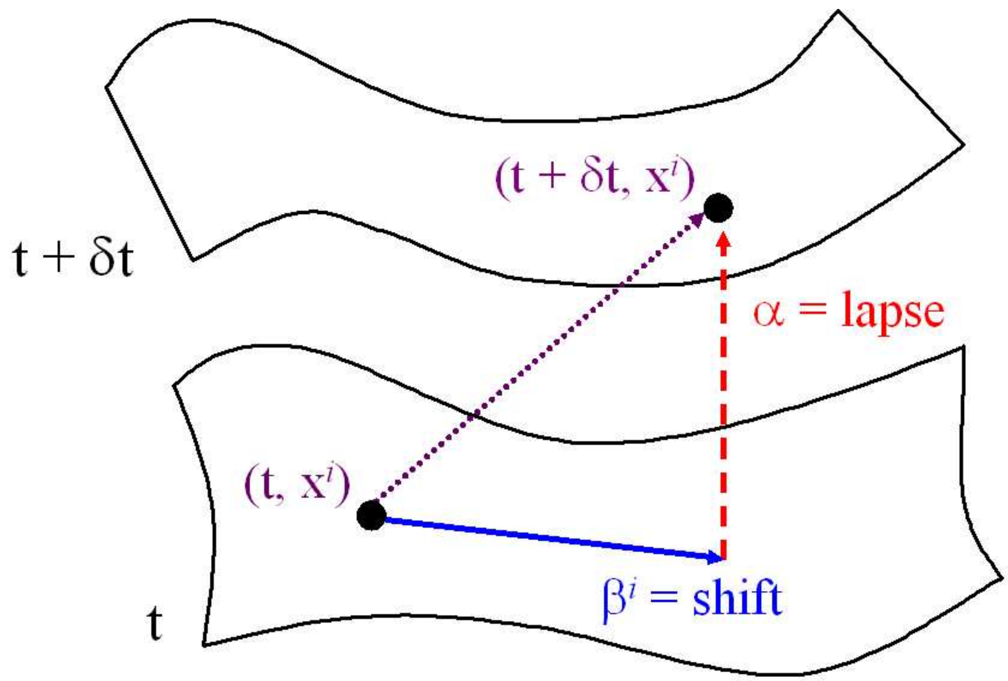

Figure 2.2: Diagram showing lapse and shift

The ADM equations in their most commonly used form are:

$$H \equiv K^2 - K_{ij}K^{ij} + R = 0 \ , \tag{2.15}$$

$$M_i \equiv D_j K_i^j - D_i K = 0 \ , \tag{2.16}$$

$$\partial_\perp g_{ij} = -2\alpha K_{ij} \ , \tag{2.17}$$

$$\partial_\perp K_{ij} = \alpha K K_{ij} - 2\alpha K_{ik} K_j^k + \alpha R_{ij} - D_i D_j \alpha \tag{2.18}$$

where $\partial_\perp \equiv \partial/\partial t - \mathcal{L}_\beta$ represents the time derivative. $\mathcal{L}_\beta$ is the Lie Derivative along β^i, and D_i is the covariant derivative. H is called the Hamiltonian constraint and M the momentum constraint. The covariant derivative for a tensor T_{ij} is defined as

$$D_k T_{ij} = \partial_k T_{ij} - \Gamma^l_{ki} T_{lj} - \Gamma^l_{kj} T_{il}, \tag{2.19}$$

and the Lie Derivative of a tensor T_{ij} with respect to a vector field β^k is

$$\mathcal{L}_\beta T_{ij} = \beta^k \partial_k T_{ij} + T_{kj} \partial_i \beta^k + T_{ik} \partial_j \beta^k. \tag{2.20}$$

Chapter 3

The Initial Value Problem

Four of the ADM equations do not contain second time derivatives of the metric; they are constraint equations. The constraints need to be solved to obtain valid initial data which can then be evolved using the evolution equations. The Hamiltonian and momentum constraint equations are

$$H \equiv K^2 - K_{ij}K^{ij} + R = 0 \tag{3.1}$$

$$M_i \equiv D_j K_i^j - D_i K = 0. \tag{3.2}$$

The constraints in the form of equation (3.1) cannot be solved uniquely for the metric and extrinsic curvature. Something must be done to put the equations into a standard form that can be solved uniquely. The equations are decomposed into freely specifiable quantities and quantities to be solved for. Section (3.1) explains the conformal decomposition, which is a common way of accomplishing this. After the conformal decomposition we are left with a set of four nonlinear elliptic partial differential equations (PDE's). Multigrid is one of the most efficient ways of solving an elliptic PDE. Section 3.2 gives an introduction to our multigrid paper. Using our multigrid solver AMRMG, we have solved for a new class of initial data: distorted black holes using the puncture method. Section 3.3 gives an introduction to our paper on distorted black holes.

3.1 The Conformal Decomposition

3.1.1 Introduction

There are 12 unknowns (g_{ij},K_{ij}) but only four constraint equations in the initial value problem. Something must be done in order to transform this set of equations into a standard elliptic system of equations with a unique solution, i.e. four equations with four unknowns. There are several different ways of doing this, and perhaps one of the most common is the York–Lichnerowicz conformal decomposition[1]. The conformal decomposition takes the system of 12 unknowns g_{ij} and K_{ij} and transforms it into a new system of only four unknowns which will be labeled ψ and w^a.

Any tensor can be decomposed into a scalar multiplier times another tensor. We call the scalar multiplier the 'conformal factor' and say that these tensors are 'conformally related'. We can decompose a tensor C^{ij} into a conformal factor times a conformal tensor (denoted with tilde) $\tilde{C}^{ij}$:

$$C^{ij} = \psi^n \tilde{C}^{ij} \tag{3.3}$$

Here, the conformal factor ψ can be to any power of n. The n is chosen in such a way that it simplifies or gives a desired form to the math in question. In the case of the metric decomposition for the initial value equations, n is chosen to be -4, so that $g^{ij} = \psi^{-4}\tilde{g}^{ij}$ and $g_{ij} = \psi^4 \tilde{g}_{ij}$.

There are further tensor manipulations that are used in the initial value decomposition. One technique is to break up a tensor into its trace and tracefree parts. Any symmetric tensor may be decomposed in this manner. For example, we can decompose a symmetric tensor C^{ij} into its trace and tracefree part (F^{ij})

$$C^{ij} = \frac{1}{3} C^k_k \delta^{ij} + F^{ij}. \tag{3.4}$$

We can further decompose C^{ij} using what is known as a transverse-traceless decomposition. Any trace-free symmetric tensor (like F^{ij}) can be split as

$$F^{ij} = B^{ij} + (Lw)^{ij} \tag{3.5}$$

[1]Living Review article [37] gives more information on initial data decompositions.

where $(Lw)^{ij} = D^i w^j + D^j w^i - \frac{2}{3} g^{ij} D_k w^k$ and B^{ij} is transverse traceless ($D_j B^{ij} = 0$ and $\mathrm{tr} B^{ij} = 0$).

To summarize, there are some tensor manipulations that can be done to rearrange the initial value equations. One can perform a conformal decomposition and a transverse traceless decomposition. The resulting equations differ depending on which order these two operations are performed and which power of conformal factor is used.

3.1.2 Conformal Transverse Traceless

The most common way of doing the initial value decomposition is the conformal transverse traceless method. This method is performed by first doing a conformal decomposition and then performing the transverse traceless decomposition of the tracefree part of the extrinsic curvature K_{ij}.

The conformal metric $\tilde{g}_{ij}$ is defined by

$$g_{ij} = \psi^4 \tilde{g}_{ij} \tag{3.6}$$

In the following, variables with a tilde (e.g.,$\tilde{g}$) are conformal variables, to be raised and lowered with the conformal metric. To simplify the momentum constraint (K equations), first we separate K^{ij} into its trace and tracefree parts. Any symmetric tensor can be decomposed into a trace and tracefree part.

$$K^{ij} = A^{ij} + \frac{1}{3} g^{ij} K \tag{3.7}$$

Now we perform a conformal decomposition of the tracefree part of K_{ij}:

$$A^{ij} = \psi^n \tilde{A}^{ij} \tag{3.8}$$

Here n can be any number, but because when $n = 10$ we have

$$D_j A^{ij} = \psi^{-10} \tilde{D}_j \tilde{A}^{ij}, \tag{3.9}$$

$n = 10$ is typically chosen as it will simplify the momentum constraint. If we let n be any number, we have

$$K^{ij} = \psi^n \tilde{A}^{ij} + \frac{1}{3} g^{ij} K. \tag{3.10}$$

Any trace-free symmetric tensor (like $\tilde{A}^{ij}$) can be split as

$$\tilde{A}^{ij} = \tilde{B}^{ij} + (\tilde{L}w)^{ij} \tag{3.11}$$

where $(\tilde{L}w)^{ij} = \tilde{D}^i w^j + \tilde{D}^j w^i - \frac{2}{3}\tilde{g}^{ij}\tilde{D}_k w^k$ and $\tilde{B}^{ij}$ is transverse traceless ($\tilde{D}_j \tilde{B}^{ij} = 0$ and $\mathrm{tr}\tilde{B}^{ij} = 0$).

So in terms of this splitting, we have

$$K^{ij} = \psi^n \tilde{B}^{ij} + \psi^n (\tilde{L}w)^{ij} + \frac{1}{3}\psi^{-4}\tilde{g}^{ij}K \tag{3.12}$$

The Hamiltonian constraint and momentum constraint become:

$$\begin{aligned} H &= -\psi^{2n+8}\tilde{A}_{ab}\tilde{A}^{ab} + \frac{2}{3}K + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi && (3.13) \\ M_a &= (n+10)\psi^{n+3}\tilde{D}_b\psi\tilde{A}^b_a + \psi^{n+4}\tilde{D}_b\tilde{A}^b_a - \frac{2}{3}\tilde{D}_a K && (3.14) \end{aligned}$$

If $n = -10$, we get the 'usual' conformal transverse traceless equations.

$$\begin{aligned} H &= -\psi^{-12}\tilde{A}_{ab}\tilde{A}^{ab} + \frac{2}{3}K + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi && (3.15) \\ M_a &= \psi^{-6}\tilde{D}_b\tilde{A}^b_a - \frac{2}{3}\tilde{D}_a K && (3.16) \end{aligned}$$

Using $n = -4$ also provides some simplification, and we get what will later be referred to as the 'same sign decomposition' (same sign as the n for the conformal inverse metric).

$$\begin{aligned} H &= -\tilde{A}_{ab}\tilde{A}^{ab} + \frac{2}{3}K + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi && (3.17) \\ M_a &= 6\psi^{-1}\tilde{D}_b\psi\tilde{A}^b_a + \tilde{D}_b\tilde{A}^b_a - \frac{2}{3}\tilde{D}_a K && (3.18) \end{aligned}$$

3.1.3 Physical Transverse Traceless

The physical transverse traceless method is when one performs a transverse traceless decomposition first and then use a conformal decomposition. As before, we start with

$$K^{ij} = A^{ij} + \frac{1}{3}g^{ij}K \tag{3.19}$$

and then break up A^{ij} into transverse and traceless parts

$$K^{ij} = B^{ij} + (Lw)^{ij} + \frac{1}{3}g^{ij}K \tag{3.20}$$

where B^{ij} is transverse traceless and $(Lw)^{ij} = D^i w^j + D^j w^i - \frac{2}{3}g^{ij}D_k w^k$.

Now we can do the conformal transformation

$$B^{ij} = \psi^n \tilde{B}^{ij} \tag{3.21}$$

We can pick any n, but $\tilde{B}$ is only transverse if we pick $n = -10$. Using the conformal metric and the corresponding $\tilde{D}$ operators we find the relation

$$(Lw)^{ij} = \psi^{-4}(\tilde{L}w)^{ij} \tag{3.22}$$

Now we have

$$K^{ij} = \psi^n \tilde{B}^{ij} + \psi^{-4}(\tilde{L}w)^{ij} + \frac{1}{3}g^{ij}K \tag{3.23}$$

The Hamiltonian constraint and momentum constraint become:

$$\begin{aligned}
H &= -(\tilde{L}w)_{ab}(\tilde{L}w)^{ab} - 2\psi^{n+4}\tilde{B}_{ab}(\tilde{L}w)^{ab} - \psi^{2n+8}\tilde{B}_{ab}\tilde{B}^{ab} && (3.24)\\
&\quad + \frac{2}{3}K^2 + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi && (3.25)\\
M_a &= (n+10)\psi^{n+3}\tilde{D}_b\psi\tilde{B}^b_a + \psi^{n+4}\tilde{D}_b\tilde{B}^b_a + \tilde{D}_b(\tilde{L}w)^b_a && (3.26)\\
&\quad + \frac{6}{\psi}\tilde{D}_b\psi(\tilde{L}w)^b_a - \frac{2}{3}\tilde{D}_aK && (3.27)
\end{aligned}$$

If $n = -10$, (and $\tilde{B}$ is transverse), we get ‘the usual’ physical transverse traceless equations:

$$\begin{aligned}
H &= -(\tilde{L}w)_{ab}(\tilde{L}w)^{ab} - 2\psi^{-6}\tilde{B}_{ab}(\tilde{L}w)^{ab} - \psi^{-12}\tilde{B}_{ab}\tilde{B}^{ab} && (3.28)\\
&\quad + \frac{2}{3}K^2 + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi && (3.29)\\
M_a &= \tilde{D}_b(\tilde{L}w)^b_a + \frac{6}{\psi}\tilde{D}_b\psi(\tilde{L}w)^b_a - \frac{2}{3}\tilde{D}_aK && (3.30)
\end{aligned}$$

3.1.4 Summary

The initial value equations are transformed by using conformal decomposition techniques. The most common method, and the method used in our initial data solver, is the conformal transverse traceless decomposition. After a conformal transverse traceless decomposition, the initial value equations are

$$H = -\psi^{-12}\tilde{A}_{ab}\tilde{A}^{ab} + \frac{2}{3}K + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi \tag{3.31}$$

$$M_a = \psi^{-6}\tilde{D}_b\tilde{A}^b_a - \frac{2}{3}\tilde{D}_a K. \tag{3.32}$$

Often the initial extrinsic curvature is chosen to be zero (this is called 'a moment of time symmetry'). This means that the momentum constraint is automatically satisfied and the Hamiltonian constraint reduces to

$$H = \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi. \tag{3.33}$$

3.2 AMR Multigrid

The Hamiltonian constraint is a nonlinear elliptic equation and must be solved numerically. Methods for solving elliptic systems of equations can be classified as direct or iterative. Gaussian elimination is an example of a direct solver. Direct solvers require a tremendous amount of storage, usually of order N^3, where N is the number of grid points. Iterative methods generally only need storage of order N. In numerical relativity, we need to store a grid large enough to encompass a vast amount of space. We also need high resolution for accuracy. The memory required for such a large, well resolved grid excludes the possibility for employing direct solvers, and iterative methods must be used. Usually, an iterative method known as 'relaxation' (such as Jacobi or Gauss–Seidel) is used. These methods are slow to converge and their performance degrades with finer grids. For a grid of N points, with grid resolution $h = O(1/N)$, the speed of convergence goes to zero as h goes to zero.

This slow rate of convergence can be explained by examining the error in terms of frequency and wavelengths. Error can be decomposed into wavelengths. Iterative methods work by 'smoothing out' the error via a relaxation scheme. A typical relaxation operator 'sweeps' over the grid, updating each point by a weighted average of its neighboring points. Figure 3.1 shows a graph of the error of a solution on a well

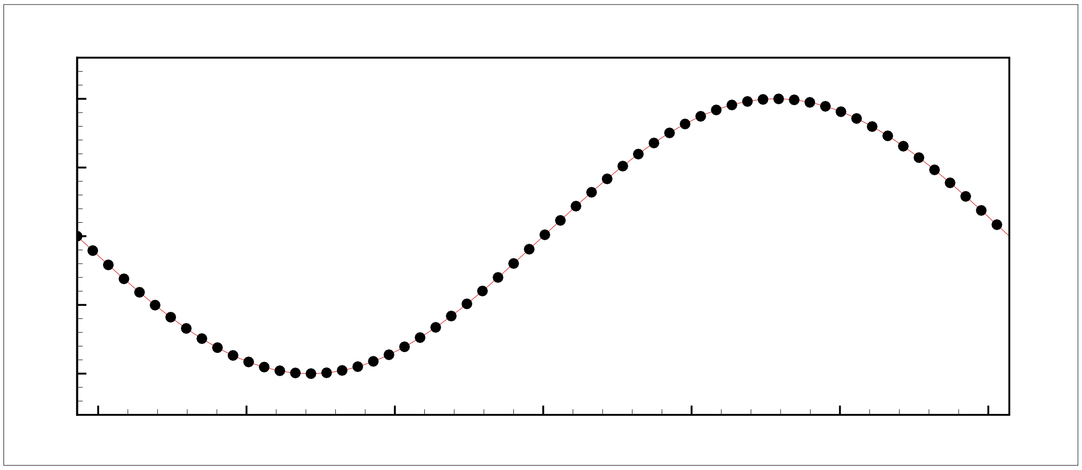

Figure 3.1: With a well resolved grid, there is little difference between values of the function at neighboring grid points and relaxation does not decrease the error efficiently.

resolved grid (many grid points). With a well resolved grid, there is little difference between values of the function at neighboring grid points and relaxation does not decrease the error efficiently. If the error already appears smooth (in the perspective of each grid point), there is no significant improvement with successive relaxation sweeps, and this results in a slow convergence rate. If we put the same function as in figure 3.1 on a coarse grid as in figure 3.2, the error is apparent and can be easily diminished through relaxation. It can be seen in figures 3.1 and 3.2 that coarse grids can be better for eliminating certain wavelengths of error, but a coarse grid cannot be used for solving a detailed problem. Multigrid uses a series of coarse and fine grids to smooth the error at all frequencies. The general idea of multigrid is to put a problem on a fine grid with a trial solution and use a relaxation operator damp the high frequency error. Then we restrict the data to a coarser grid and relax again, damping the low frequency error. Then we can prolong the solution back onto the finer grid for a final solution[2].

Even with a memory efficient iterative method, the resources needed to solve a three-dimensional relativity problem can be prohibitive. Great detail (via a great many grid points) is needed to resolve sources such as black holes. In order to capture

[2]See reference [38] for tutorial on multigrid methods.

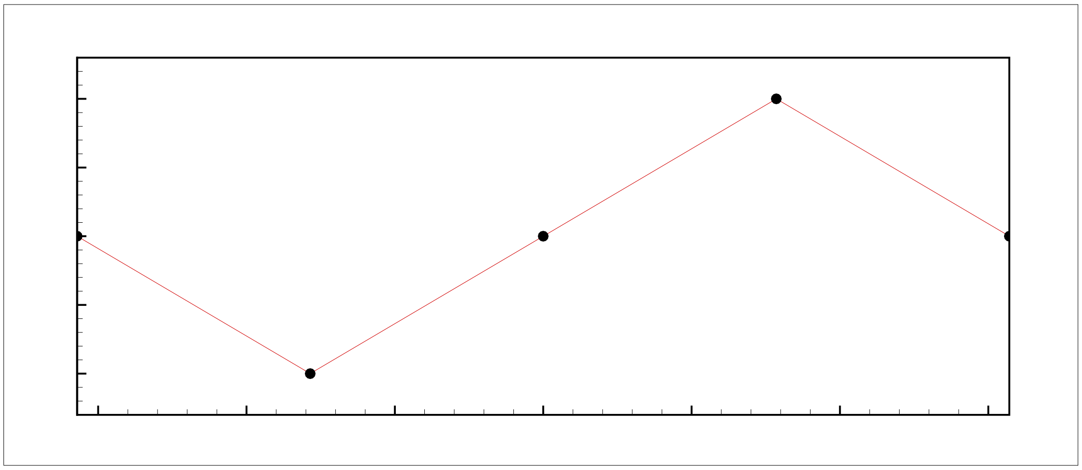

Figure 3.2: If we put a function on a coarse grid, the error is apparent and can be easily diminished through relaxation.

gravitational waves and ensure accurate boundary conditions, a very large domain is needed. It is not computationally possible to use high resolution everywhere for such an enormous three-dimensional grid. Adaptive mesh refinement (AMR) is a way of making a grid that has high resolution only where necessary. The disparity in length scales in general relativity makes AMR essential. Gravitational wave length scales can be 100 times the length of a source.

AMR saves a great deal of computational resources by reducing the number of grid points needed, but the problem is still very intensive. The problem must be solved on large computers using parallel algorithms. We use a package called PARAMESH to handle the parallel infrastructure and communications in our multigrid code [39, 40][3]. An AMR grid is established by refining parts of the grid with a relative truncation error higher than a specified tolerance. In the testing process, sometimes it is more instructive to have a fixed grid. Figure 3.3 shows an example of a grid that uses fixed mesh refinement (FMR). In this case, a certain grid structure will be referred to as an 'XpY' or 'X plus Y' grid. This means that PARAMESH has refined X times for its base grid, and then has Y more refined grids on top of that. Figure 3.3 shows a '3p2' grid

[3]The PARAMESH software used in this work was developed at the NASA Goddard Space Flight Center and Drexel University under NASA's HPCC and ESTO/CT projects and under grant NNG04GP79G from the NASA/AISR project.

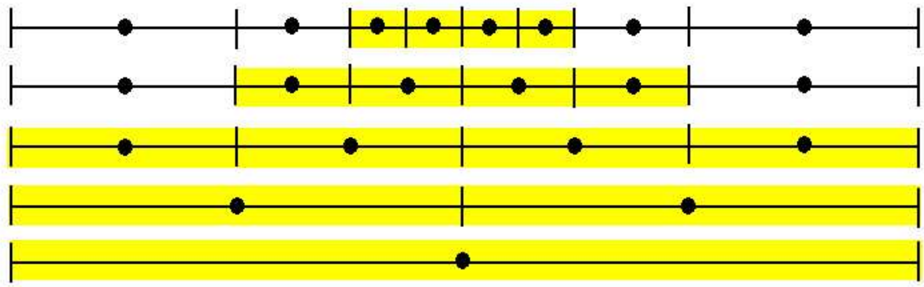

Figure 3.3: A PARAMESH 3p2 grid hierarchy. The Multigrid solver will cycle over the highlighted levels.

and its underlying grid hierarchy. The Multigrid solver will cycle over the highlighted levels. Some multigrid methods, given a final grid like the top of figure 3.3, will begin with an initial solution and perform relaxation along the entire top level, and then restrict down to the next level also relaxing the error along this entire next level, and so on. This is known as FAC, for Fast Adaptive Composite grid [41]. Our multigrid does not use FAC, rather it cycles only over the highlighted portion of figure 3.3, beginning with only the highest resolution part of the grid (the highlighted part of the top level in figure 3.3), restricting the data to the blocks immediately underneath, then performing relaxation on the next highest resolution part of the grid, and so on. This is known as MLAT, or Multi Level Adaptive Technique [42]. FAC may have an advantage of providing smoother solutions, but the MLAT approach saves a great deal of time and space by only defining and cycling over necessary portions of the grid.

Chapter 4 is the paper that explains the specifics of our code AMRMG(Adaptive Mesh Refinement Multigrid) in detail. Because of limited space, we were not able to show many interesting figures of test cases in the paper, so a few are included here. Recall the Hamiltonian constraint equation

$$H = -\psi^{-12}\tilde{A}_{ab}\tilde{A}^{ab} + \frac{2}{3}K + \psi^{-4}\tilde{R} - 8\psi^{-5}\tilde{D}^2\psi. \tag{3.34}$$

Given a conformal metric $\tilde{g}_{ij}$ and extrinsic curvature[4] $\tilde{K}_{ij}$, the multigrid code solves

[4]Currently, AMRMG only solves the Hamiltonian constraint given an analytic form for K_{ij}. It

the Hamiltonian constraint for the conformal factor ψ. Initial data with an arbitrary number of black holes can be generated using what is known as 'the puncture method', which is described in further detail in the next section. The puncture method separates the conformal factor into singular and nonsingular parts as $\psi = u + \frac{m}{2r}$. What is shown in the graphs is u, the nonsingular part of the metric. Figures 3.4 and 3.5 show contour plots of u for initial data with two equal mass black holes, and figure 3.6 shows the corresponding AMR grid structure as determined adaptively by AMRMG. To further illustrate the utility of adaptive mesh refinement, figures 3.7 and 3.8 show initial data for seven black holes of various position, mass, and momentum and its corresponding mesh. All data sets are three-dimensional (3D); the plots show a 2D cut of the data taken at $x = 0$ for the two black hole case and at $z = 0$ for the seven black hole case.

3.3 Distorted Black Holes

A distorted black hole is a black hole with the addition of some gravitational waves. It is an attempt to create more realistic, astrophysically relevant initial data. A distorted black hole metric may mimic a spacetime directly following the merger of two black holes. The first distorted black hole metric in axisymmetry was proposed by Bernstein, et. al. [43]. This metric was subsequently modified for the general, non-symmetric case [44]. The black hole boundary condition was implemented by using an isometry condition. We have adapted the problem so that there does not have to be an isometry, but rather we use the puncture method to implement the inner boundary conditions [5]. The difficulty lies then with having enough resolution around the puncture, which makes our AMR very useful. Also, by using the puncture method, one can have multiple black holes, whereas this is not possible using isometry.

A distorted black hole metric starts out as the metric for a Brill wave [45], i.e., strong gravitational waves:

$$ds^2 = \psi^4[e^{2q}(dr^2 + r^2\, d\theta^2) + r^2 \sin^2\theta\, d\phi^2]\ , \tag{3.35}$$

---

does not solve for the momentum constraint, although in principle it could be modified to do so.

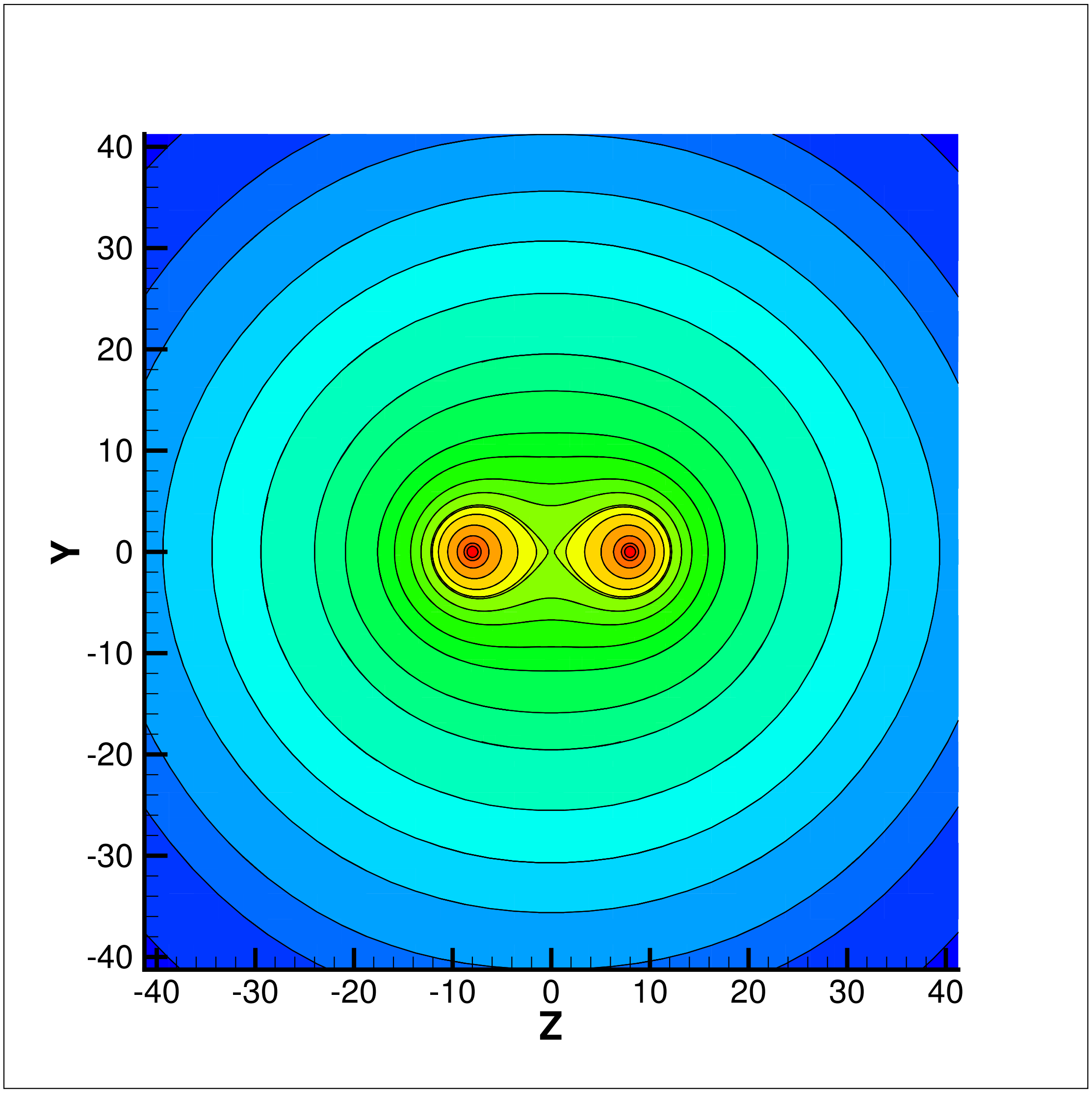

Figure 3.4: 2D contour plot of two black holes using puncture data and AMR.

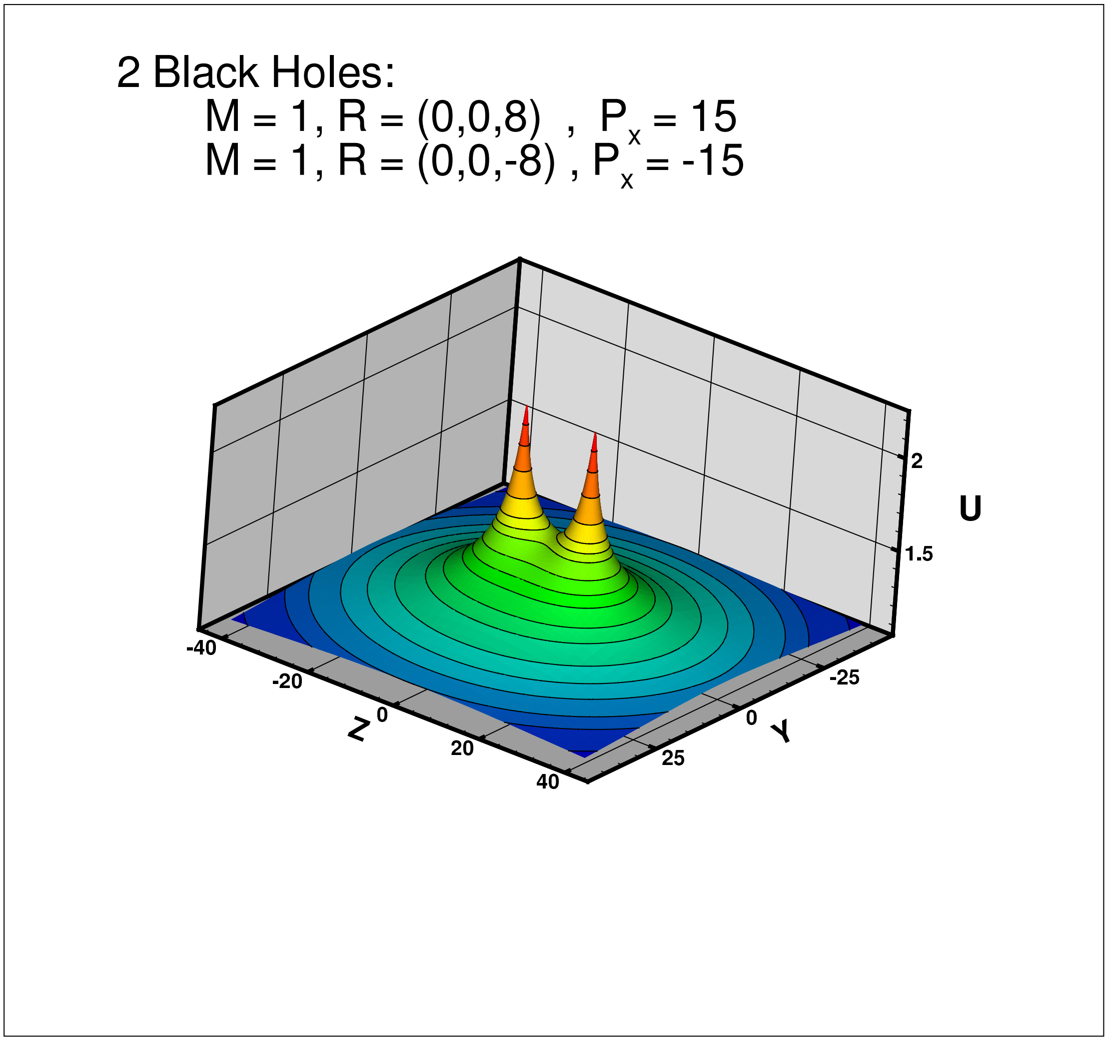

Figure 3.5: 3D contour plot of two black holes using puncture data and AMR. M is mass, R is location of the black hole, and P_x is x component of the momentum.

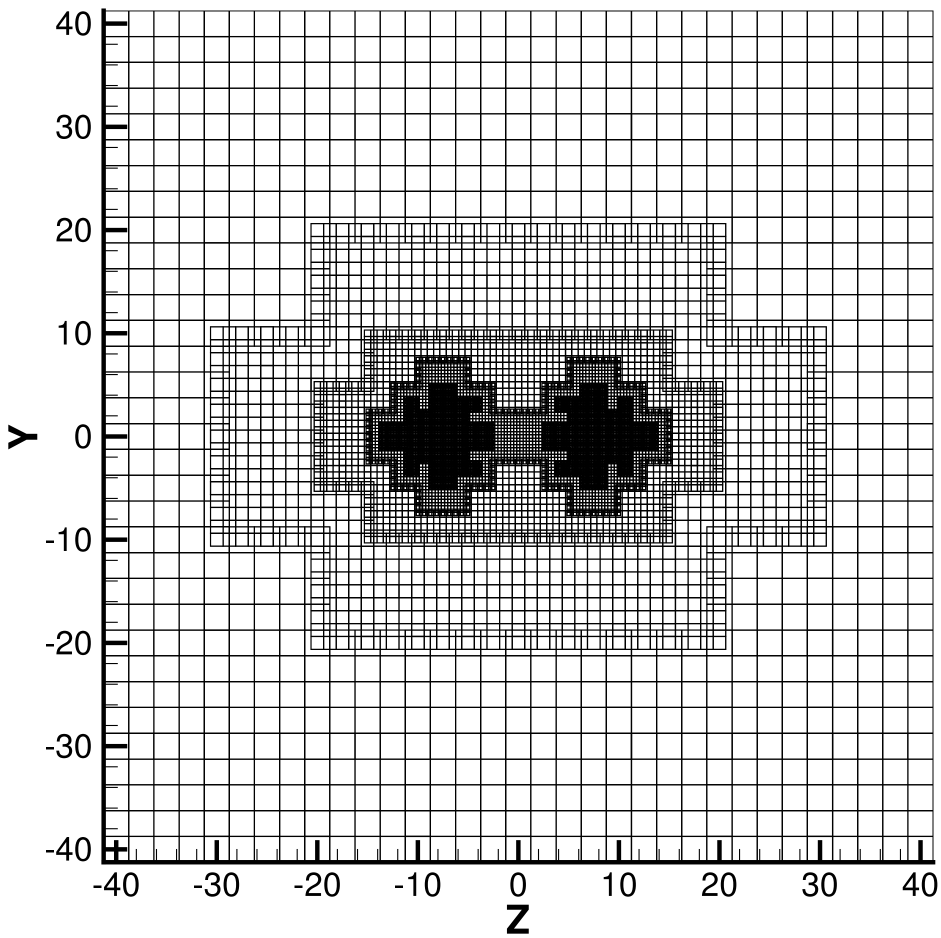

Figure 3.6: Mesh for two black holes using puncture data and AMR

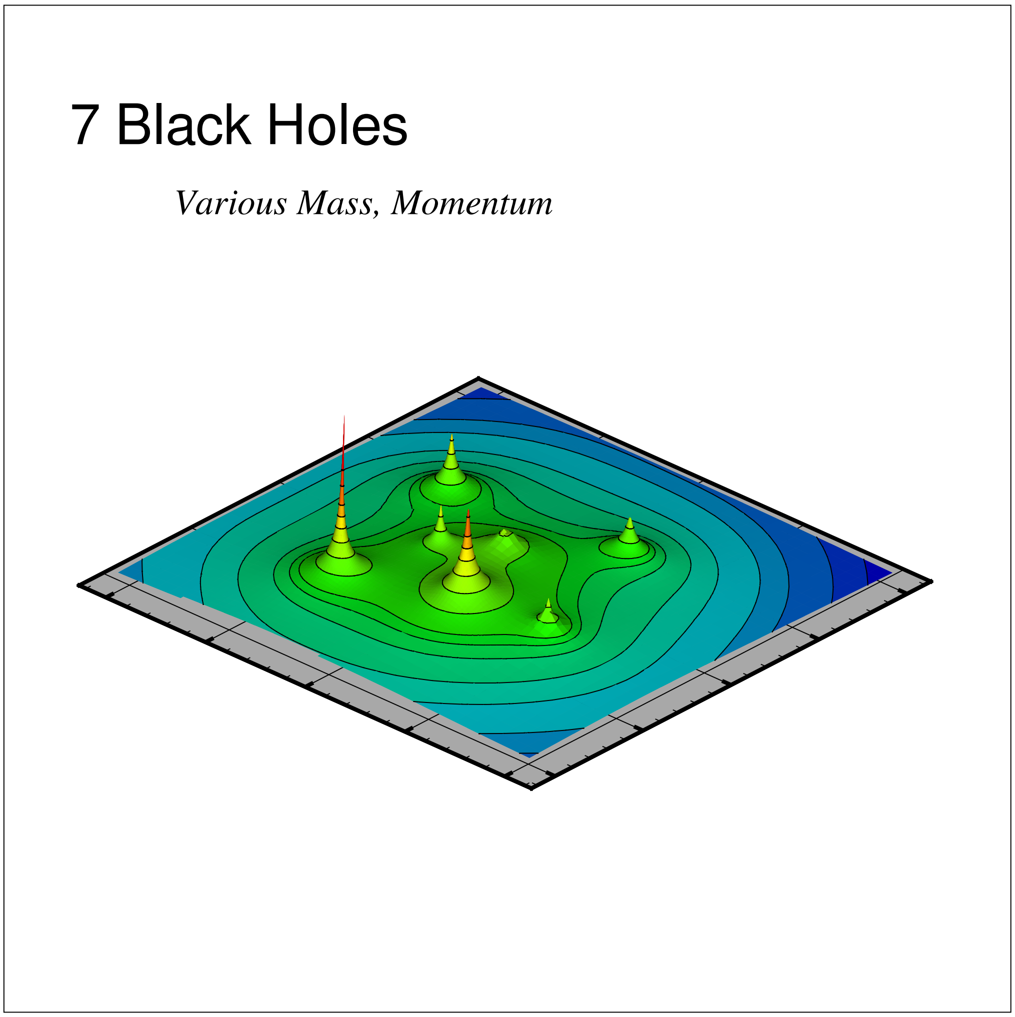

Figure 3.7: Seven black holes using puncture data and AMR. Shown is u, the nonsingular part of the conformal factor ψ, for seven black holes with various mass, momenta and locations.

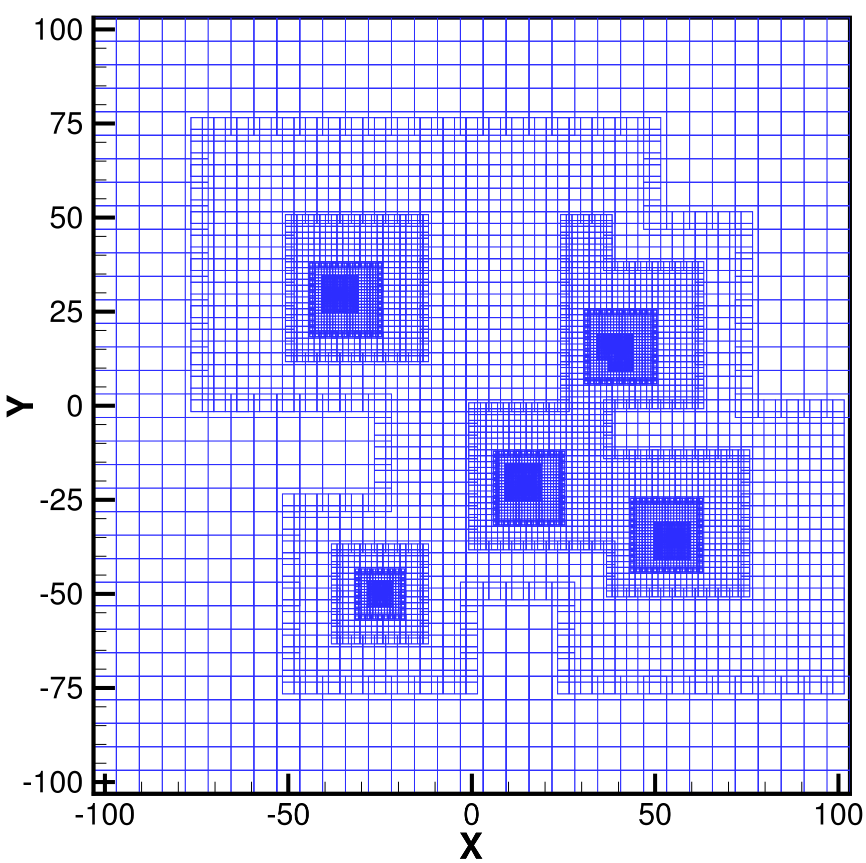

Figure 3.8: Mesh for seven black holes using puncture data and AMR

where q is a specified function of the spatial coordinates and ψ is the conformal factor. We also restrict the data to a moment of time symmetry so that $K = 0$ and the momentum constraints are satisfied. The Hamiltonian constraint is then

$$\tilde{\nabla}^2\psi - \frac{1}{8}\tilde{R}\psi = 0 \ , \tag{3.36}$$

where $\tilde{\nabla}^2$ and $\tilde{R}$ are the Laplacian and scalar curvature of the conformal metric $\tilde{g}_{ij}$, defined by $g_{ij} = \psi^4\tilde{g}_{ij}$. The outer boundary condition is the ordinary Robin condition $[\partial(r\psi - r)/\partial r = 0$ as $r \to \infty]$. This initial data so far presented gives a Brill wave. To add a black hole, we must impose 'inner boundary conditions', which are boundary conditions indicating where the black holes are. Bernstein, et. al. [43] used an isometry condition. Isometric means symmetric under reflections about the black hole throat. To implement this, they choose a function q that satisfies an isometry condition. Instead of using isometry for the black hole boundary condition, we use a puncture. The puncture method [19] is a way of making a black hole inner boundary condition by imposing the conformal factor to be in the form of $\psi = u + \frac{m}{2r}$. This is inserted into the Hamiltonian constraint (3.36) to obtain an elliptic equation in u:

$$\tilde{\nabla}^2 u - \frac{1}{8}\tilde{R}u = \frac{m}{16r}\tilde{R} \ . \tag{3.37}$$

For certain choices of parameters in the function q, our data sets are theoretically equivalent to those of Brandt et. al. [44]. To verify the equivalence (and thus validity) of our puncture data, we set the required parameters (see details in chapter 5) to get equivalent data sets, and compared our calculated ADM masses with those in [44]. We produced a figure equivalent to figure 3 of [44] and compared the two figures, finding them visually identical. Figure 3.9 shows our data for the ADM masses. We also include the data for the ADM masses at the punctures. For isometric data sets, the ADM mass at infinity and the ADM mass at the puncture (which is at conformal infinity) should be equal. Note that in the figure the two ADM masses are not exactly equal. We have verified that at higher resolutions the values for the ADM masses approach equality.

By using the puncture method, we can create distorted black hole initial data sets formerly solved using an isometry condition. Additionally, we can create new sets

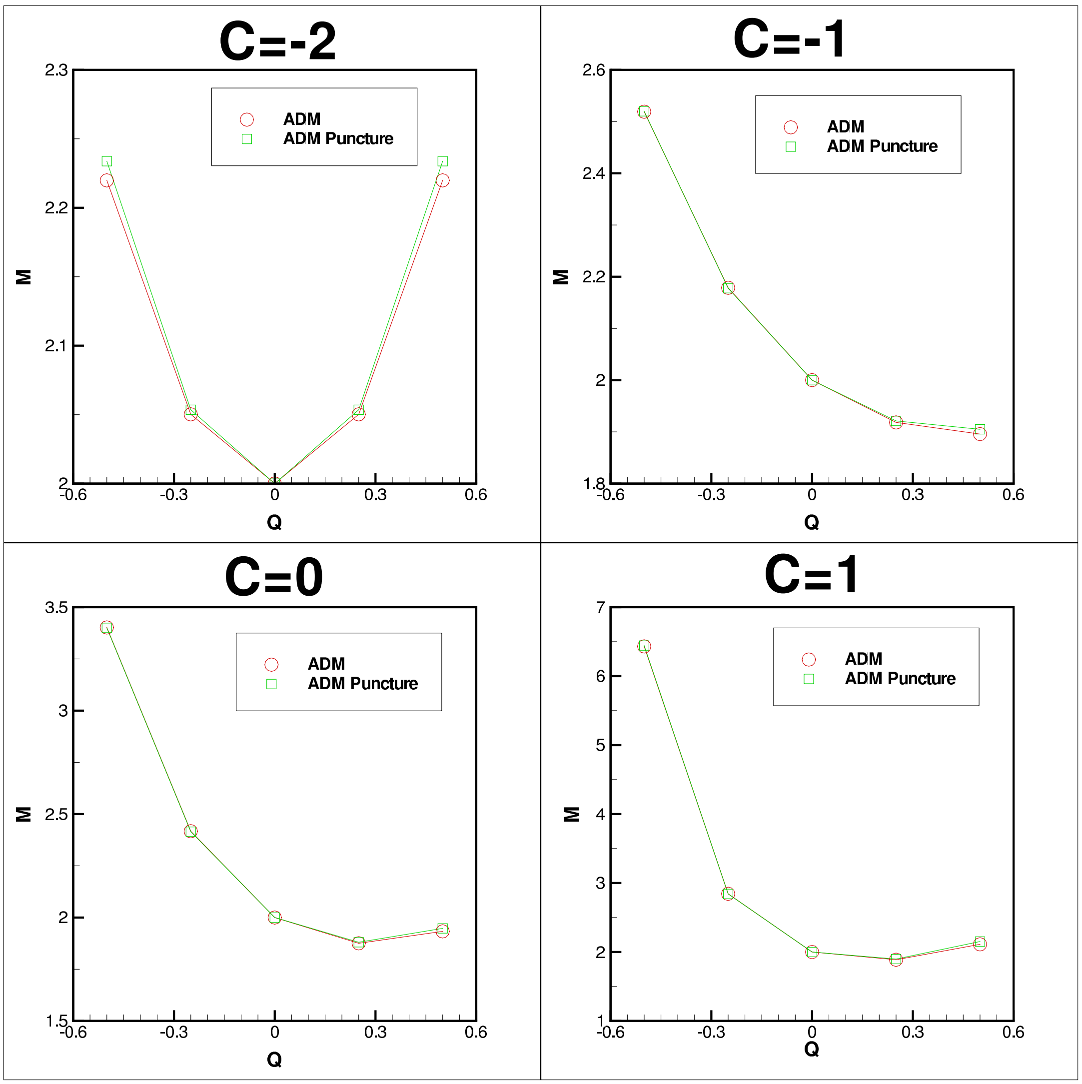

Figure 3.9: The ADM masses at infinity and at the punctures (conformal infinity) from our data of distorted black holes using the puncture method [5]. Data points are calculated in a way to enable comparison with figure 3 of [44].

of initial data by using parameters different from those resulting in isometry. Since these are original data sets, we need some check for validity other than comparison. As a check for the validity of our data, we performed convergence tests of the ADM masses. Table 3.1 shows the results of a test that keeps the outer boundary fixed and increases the grid resolution. The convergence rate confirms second order convergence of the ADM masses. Table 3.2 shows the results of a test that kept a fixed resolution while increasing the outer boundary. The test shows that the ADM masses do not vary greatly with outer boundary location.

Table 3.1: The results of a test that keeps the outer boundary fixed and increases the grid resolution. Column A is the ADM Mass at Infinity and B is the ADM Mass of the puncture. Columns C and D show the difference in ADM masses between the value at the same row and the value at the next higher resolution. Columns E and F divide the difference value with the value at the next highest resolution to obtain a convergence rate. The convergence rate confirms second order convergence of the ADM masses.

Type of Grid	Convergence Data of the ADM Masses for a Fixed Outer Boundary					
	A	B	C	D	E	F
3p2	2.27354	2.31592	0.04107	0.08214	3.4600	3.4600
4p2	2.21985	2.23378	0.01187	0.02374	3.9699	3.9699
5p2	2.20648	2.21004	0.00299	0.00598		
6p2	2.20317	2.20406				

Table 3.2: The results of a test keeping a fixed resolution while increasing the outer boundary. The test shows that the ADM masses do not vary greatly with outer boundary location.

Type of Grid	Convergence Data of the ADM Masses for a Fixed Resolution	
	ADM Mass at Infinity	ADM Mass at Puncture
5p3	2.206455	2.210016
5p4	2.206445	2.210016
5p5	2.206435	2.210014

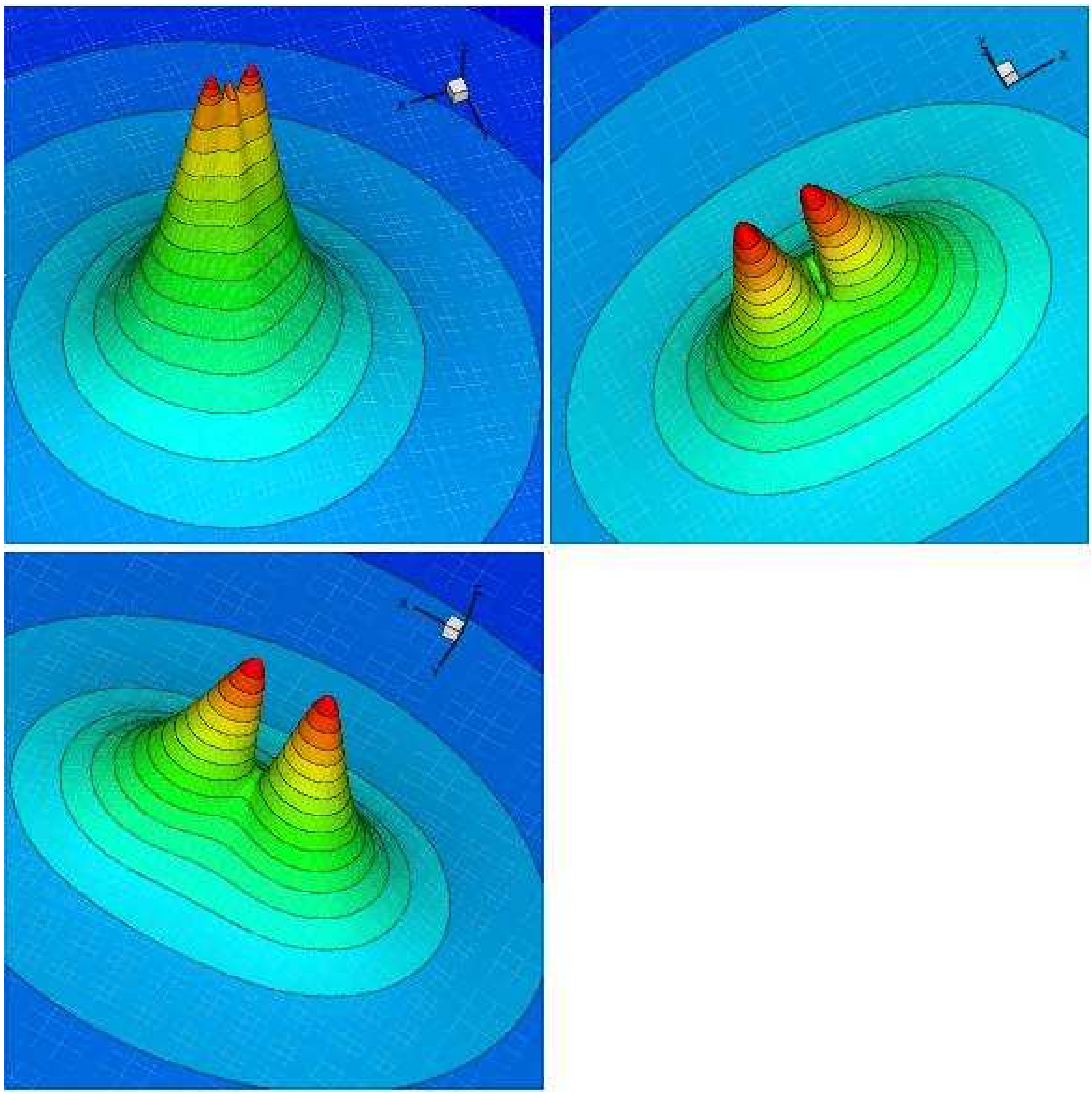

Figure 3.10: Distorted black holes using the puncture method. Shown are 2D contour plots in the X, Y, Z planes of u, the nonsingular part of the conformal factor ψ.

Chapter 4

Multigrid elliptic equation solver with adaptive mesh refinement

J. David Brown and Lisa L. Lowe
Multigrid elliptic equation solver with adaptive mesh refinement
Journal of Computational Physics

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 209 (2005) 582–598

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Multigrid elliptic equation solver with adaptive mesh refinement

J. David Brown, Lisa L. Lowe *

Department of Physics, North Carolina State University, Raleigh, NC 27695, USA

Received 24 November 2004; received in revised form 9 March 2005; accepted 18 March 2005
Available online 25 May 2005

Abstract

In this paper, we describe in detail the computational algorithm used by our parallel multigrid elliptic equation solver with adaptive mesh refinement. Our code uses truncation error estimates to adaptively refine the grid as part of the solution process. The presentation includes a discussion of the orders of accuracy that we use for prolongation and restriction operators to ensure second order accurate results and to minimize computational work. Code tests are presented that confirm the overall second order accuracy and demonstrate the savings in computational resources provided by adaptive mesh refinement.

1. Introduction

Elliptic equations appear throughout engineering, science, and mathematics. Our primary interest is in elliptic problems that arise in the context of numerical relativity. Currently, the field of numerical relativity is being driven by rapid progress on the experimental front. There are several ground-based gravitational wave detectors in operation today, and their sensitivities are quickly approaching a level at which interesting science can be done. There are also plans for a space-based gravitational wave detector, LISA, to be launched around 2012. The scientific payoff of these instruments will depend largely on our ability to theoretically predict and explain the observed signals. For both ground-based and space-based gravitational wave detectors, the most common and strongest signals are expected to come from colliding black holes. Thus, much of the numerical relativity community has directed its efforts toward modeling binary black hole systems.

* Corresponding author. Tel.: +1 919 515 4196.
E-mail address: lllowe@ncsu.edu (L.L. Lowe).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2005.03.026

When black holes spiral together and collide, they generate gravitational waves. The black hole "source" region has a length scale of GM/c^2, where G is Newton's constant, M is the total mass of the two black holes, and c is the speed of light. The gravitational waves produced by the source have a length scale up to $\sim 100GM/c^2$. Herein lies one of the challenges of modeling binary black hole systems with finite difference methods. The source region requires grid zones of size $\lesssim 0.01GM/c^2$ to accurately capture the details of the black holes' interaction, while the extent of the grid needs to be several hundred GM/c^2 to accurately capture the details of the gravitational wave signal. Many research groups in numerical relativity are starting to use adaptive mesh refinement (AMR) techniques to deal with this discrepancy in length scales [1–11]. With AMR the grid resolution is allowed to vary across the computational domain so that computational resources can be concentrated where they are most needed. For the binary black hole problem, we need a high resolution region to cover the small-scale detail of the source, but the gravitational waves far from the source can be modeled with sufficient accuracy using a much lower resolution grid.

Elliptic equations occur in several contexts in numerical relativity. Einstein's theory of gravity is a system of partial differential equations consisting of four constraint equations and a set of evolution equations (see for example [12]). The constraint equations restrict the data at each time step so in particular the initial data cannot be chosen freely. With suitable assumptions about the nature of the initial data, the constraint equations can be written as an elliptic system [13].

Having solved the constraints for the initial data, those data are evolved forward in time by the evolution equations. At an analytical level, the evolution equations guarantee that the constraint equations will continue to be satisfied. However, in numerical modeling, numerical errors will introduce violations of the constraints. These violations can be disastrous because the evolution equations admit unphysical, constraint violating solutions that grow exponentially [14–17]. One possible strategy for preventing this disaster is to impose the constraints during the evolution, which means solving the elliptic constraint equations after each time step [18–20].

Elliptic equations also arise in numerical relativity when one is faced with choosing a coordinate system. In Einstein's theory the coordinate system must be chosen dynamically as the gravitational field evolves forward in time. The choice of coordinate system can have a dramatic effect on the performance of a numerical relativity code. Researchers have developed many different strategies for choosing a coordinate system. Some of these strategies require the solution of elliptic, parabolic or hyperbolic equations, and some involve algebraic conditions. Some researchers feel that the elliptic conditions might be best, but the cost of solving elliptic equations at each time step has made the other choices more practical and more popular.

In the numerical relativity community we need the capability of solving elliptic equations quickly on adaptive, non-uniform grids. No doubt this same need exists in other areas of science and applied mathematics.

Multigrid methods originated in the 1960s with the work of Fedorenko and Bakhvalov [21–23]. They were further developed in the 1970s by Brandt [24,25], and are now the preferred methods for solving elliptic partial differential equations. The advantage of multigrid is its speed – multigrid algorithms only require order N^3 operations to solve an elliptic equation, where N^3 is the number of grid points. In this paper, we describe our code, AMRMG, which solves nonlinear elliptic equations using multigrid methods with adaptive mesh refinement. The idea of combining multigrid with AMR is not new [24–27], although there are a number of features of our code that distinguish it from the discussions we have seen. In particular, AMRMG uses cell-centered data, as opposed to node centered data. AMRMG uses the full approximation storage (FAS) algorithm, and therefore is not restricted to linear elliptic equations. AMRMG uses the Paramesh package to implement parallelization and to organize the multigrid structure [28,29]. In [30], we used AMRMG to solve numerically for distorted black hole initial data.

AMRMG is currently set up to solve second order equations that are semi-linear (the second order derivative terms are linear in the unknown field). The FAS scheme is applicable for fully nonlinear equations as well, and in principle AMRMG can be modified to solve any nonlinear equation. The equations of current interest for us are semi-linear, therefore we have not tested AMRMG on any fully nonlinear systems.

In this paper, we describe the algorithm behind AMRMG in detail. In Section 2, we present the overall conceptual framework behind our code, and discuss some of the choices made in its development. Section 3 is devoted to a discussion of guard cell filling, which determines the coupling between fine and coarse grid regions. In Section 4, we review the FAS algorithm and in Section 5, we discuss in detail the restriction and prolongation operators used by AMRMG. In Section 6, we describe the calculation of the relative truncation error and how it is used to control the grid structure. Section 7 contains the results of a number of code tests involving the calculation of initial data for numerical relativity.

2. Multigrid with AMR

The simplest technique for solving an elliptic problem is relaxation. The equation (or system of equations) is written in discrete form as $f_i(\phi) = g_i(\phi)$, where i labels the grid points and ϕ_i denotes the numerical solution. A "relaxation sweep" consists of refining the approximate solution $\phi_i^{\rm old}$ by solving the system $f_i(\phi^{\rm new}) = g_i(\phi^{\rm old})$ for $\phi_i^{\rm new}$. [In the simplest case f is the identity and relaxation is written as $\phi_i^{\rm new} = g_i(\phi^{\rm old})$.] The success of the relaxation method depends on how the finite difference equations are split into a left-hand side $f_i(\phi)$ and a right-hand side $g_i(\phi)$. When relaxation does work, it is slow to converge. In particular the long wavelength features of the solution must slowly "diffuse" across the grid with successive relaxation sweeps.

To solve a problem with multigrid methods we introduce a hierarchy of grids with different resolutions. For the moment, consider the case in which we seek the numerical solution of an elliptic equation on a uniform grid of size N^3 that covers the entire computational domain. We introduce grids of size $(N/2)^3$, $(N/4)^3$, etc., each covering the computational domain. On each grid the finite difference equation, or an associated equation, is solved by relaxation. The equations to be solved on each multigrid level are discussed in Section 4. For now, we simply note that the equations are chosen so that relaxation on the coarse grids quickly captures the long wavelength features of the solution. Relaxation on the fine grids captures the short wavelength features. The grids in the multigrid hierarchy communicate with one another through restriction and prolongation operators. Restriction takes data on a grid in the hierarchy and restricts it to the next coarsest grid. Prolongation takes data on a grid in the hierarchy and interpolates it onto the next finest grid. Different multigrid algorithms use different sequences of grids in solving elliptic problems, but the most basic sequence is the V-cycle. In a multigrid V-cycle one starts with the finest grid, steps down the grid hierarchy to the coarsest grid, then steps back up to the finest grid.

In the context of a time-dependent problem, adaptive mesh refinement (AMR) means that the grid structure adapts in time to meet the changing demands as the fields evolve. In the context of an elliptic (time-independent) problem, AMR means that the grid structure is determined adaptively, as part of the solution process, in an attempt to minimize numerical errors.

We use the Paramesh package to organize the grid structure for our code. Paramesh covers the computational domain with blocks of data of varying spatial resolution. These blocks form a tree data-structure. They are logically Cartesian, consisting of a fixed number of cells. We typically use 8^3 cells for each block. Fig. 1 shows an example one-dimensional grid. The numbers in that figure indicate the resolution level, and the letters denote blocks. At the base of the tree structure is a single block, labeled 1A. Paramesh refines blocks by bisection in each coordinate direction. In this one-dimensional example block 1A is refined into two blocks, 2A and 2B. Since each data block contains the same number of cells, level 2 has twice the resolution as level 1. Using Paramesh terminology, block 1A is the "parent" of blocks 2A and 2B, and blocks 2A and 2B are the "children" of block 1A. In Fig. 1 Paramesh has

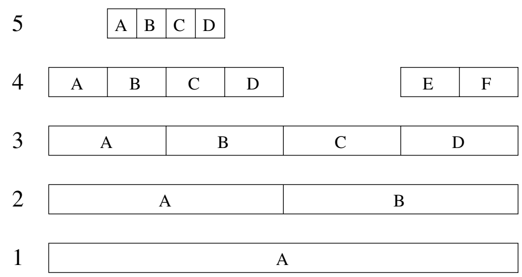

Fig. 1. Example of a one-dimensional grid structure. The numbers on the left denote the resolution level, and the letters label blocks of data.

also refined blocks 2A and 2B to create blocks 3A, 3B, 3C and 3D. Further refinements yield the non-uniform grid shown in the figure. Paramesh always creates grid structures in which adjacent blocks' refinement levels differ by no more than one.

Our first task is to decide how to carry out a basic multigrid V-cycle on a non-uniform grid structure. There are two natural approaches. The first approach is the fast adaptive composite grid method (FAC) developed by McCormick [31]. In this approach, the relaxation sweeps extend across the entire computational domain, and one defines the succession of multigrid levels by restricting the highest resolution subgrid to the next lower resolution. As an example based on Fig. 1, we let the top multigrid level consist of blocks 4A–5A–5B–5C–5D–4D–3C–4E–4F. After carrying out a series of relaxation sweeps on this non–uniform grid, we step down the multigrid V-cycle by restricting the data in blocks 5A–5B–5C–5D to resolution level 4. Thus, the next multigrid level is defined by blocks 4A–4B–4C–4D–3C–4E–4F. After relaxing on this grid, we restrict the resolution level 4 blocks to resolution level 3. This defines the next multigrid level as 3A–3B–3C–3D. We can continue in this fashion to define a complete hierarchy of multigrid levels, each covering the entire computational domain.

The second approach, the one we use for AMRMG, is to define the grids in the multigrid hierarchy to coincide with the different resolution levels. This is the original multi-level adaptive technique (MLAT) proposed by Brandt [25,32]. As an example based on Fig. 1, the top multigrid level consists of all blocks at resolution level 5, namely, 5A–5B–5C–5D. After carrying out a series of relaxation sweeps on the level 5 blocks, we restrict that data to level 4. Then the next multigrid level consists of blocks 4A–4B–4C–4D–4E–4F. After relaxing on these blocks we restrict the resolution level 4 data to resolution level 3. Then the next multigrid level consists of the level 3 blocks 3A–3B–3C–3D. We continue in this fashion to define a complete multigrid hierarchy.

We have built and tested a one-dimensional multigrid code based on the FAC approach. That code works quite well. However, the MLAT approach appeared to us to be more straightforward to implement in a three-dimensional code based on Paramesh. For this reason AMRMG defines the levels in the multigrid hierarchy by resolution. Apart from the issue of implementation, the MLAT approach has an advantage in solving problems in which only a small region of the computational domain requires high resolution. With the FAC approach, in which relaxation always extends across the entire computational domain, a lot of unnecessary computational effort can be expended on relaxation in the low resolution regions. On the other hand, the FAC approach has the advantage over MLAT in maintaining a tighter coupling between regions of different resolutions. For example, for the grid shown in Fig. 1, the data in blocks 4A and 4D effectively provide boundary conditions for relaxation in blocks 5A through 5D. With FAC, that boundary information is updated between every relaxation sweep across the top multigrid level (4A–5A–5B–5C–5D–4D–3C–4E–4F). With MLAT, in which the top multigrid level consists of blocks 5A–5B–5C–5D, the boundary information is only updated once each V-cycle.

3. Guard cell filling

AMRMG uses cell centered data. When we apply the relaxation formula $f_i(\phi^{\text{new}}) = g_i(\phi^{\text{old}})$ to a cell adjacent to a block face, the finite difference stencil extends beyond the block. Paramesh uses layers of guard cells surrounding each block to hold data from beyond the block boundaries. In Fig. 2, we show a portion of a one-dimensional grid with a fine grid block on the left and a coarse grid block on the right. The grid points are labeled by their distance from the block interface in units of the fine grid cell size Δx. Thus, the fine grid points are $-1/2$, $-3/2$, etc., and the coarse grid points are 1, 3, etc. The gray circle at location $+1/2$ is a guard cell for the fine grid block, and the gray square at location -1 is a guard cell for the coarse grid block. Guard cell values are obtained by interpolation from surrounding interior data points. We want to consider how errors in guard cell filling affect the accuracy of the solution.

Consider the simple example of the Poisson equation in one-dimension, $\partial^2\phi/\partial x^2 = \rho$. For the moment let us consider a uniform numerical grid with grid spacing Δx. With standard second order centered differencing, the discrete Poisson equation is

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} + \mathcal{O}(\Delta x^2) = \rho_i, \tag{1}$$

where i labels the grid points. The term $\mathcal{O}(\Delta x^2)$ is the truncation error obtained from discretization of the second derivative. We can rewrite Eq. (1) in a form appropriate for relaxation as

$$\phi_i^{\text{new}} = \frac{1}{2}(\phi_{i+1}^{\text{old}} + \phi_{i-1}^{\text{old}}) - \frac{1}{2}\Delta x^2 \rho_i + \mathcal{O}(\Delta x^4). \tag{2}$$

When we apply this relaxation formula, the truncation error dictates that the numerical solution will have errors of order Δx^2. That is, the numerical solution will be second order accurate.

For the non-uniform grid of Fig. 2, the discrete equation (1) and the relaxation formula (2) apply as shown in the fine grid region, where $i = -1/2, -3/2$, etc. For relaxation at the grid point $i = -1/2$, we need the guard cell value $\phi_{1/2}$. We want the guard cell value to be sufficiently accurate that it does not spoil the second order convergence of the solution. It is clear from these equations that errors of order Δx^4 in the value of $\phi_{1/2}$ can be absorbed into the truncation error already present. Thus, we expect the numerical solution to be second order convergent if the guard cells are filled to fourth (or higher) order accuracy.

As far as we know, fourth order guard cell filling is *sufficient* to produce a second order accurate solution for second order partial differential equations discretized on a non-uniform grid with standard second order differencing. Fourth order guard cell filling, however, is not a *necessary* condition. There is a "rule of thumb" in the computational mathematics community that can be summarized as follows [32,33,26]: errors of order Δx^p that occur on a subspace of dimension m in a space of dimension n will often contribute to the solution like errors of order Δx^{p+n-m} from the bulk. Thus, we anticipate that errors of order Δx^3 in guard

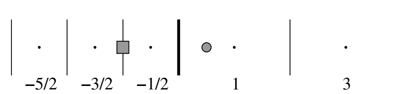

Fig. 2. A portion of a one-dimensional grid, showing three cells from a fine grid block on the left and two cells from a coarse grid block on the right.

cell filling, which occur on the two-dimensional block boundaries in the three-dimensional space, will contribute like errors of order Δx^4 from the bulk and will not spoil the second order convergence of our code. For the problems that we have studied this is indeed the case.

The guard cell filling scheme that we use was written by Kevin Olson as part of the standard Paramesh package. The process of filling the guard cells of a fine grid block that is adjacent to a coarse grid block proceeds in two steps. The first step is a restriction operation in which cells from the interior of the fine grid block are used to fill the interior cells of the underlying "parent" block. The restriction operation is depicted for the case of two spatial dimensions in the left panel of Fig. 3. The restriction proceeds as a succession of one-dimensional quadratic interpolations, and is accurate to third order in the grid spacing. Note that the fine grid stencil used for this step (nine black circles in the figure) cannot be centered on the parent cell (gray square). In each dimension the stencil includes two fine grid cells on one side of the parent cell and one fine grid cell on the other. The stencil is always positioned so that its center is shifted toward the center of the block (assumed in the figure to be toward the upper left). This ensures that only interior fine grid points, and no fine grid guard cells, are used in this first step.

For the second step, the fine grid guard cells are filled by prolongation from the parent grid. Before the prolongation, the parent block gets its own guard cells (black squares in the right panel of Fig. 3) from the neighboring block at the same refinement level. The stencil used in the prolongation operation is shown in the right panel of Fig. 3. The prolongation operation proceeds as a succession of one-dimensional quadratic interpolations, and is third order accurate. In this case, the parent grid stencil includes a layer of guard cells (black squares), as well as its own interior grid points (gray squares). At the end of this second step the fine grid guard cells are filled to third order accuracy.

When Paramesh fills the guard cells of a parent block, it also fills the guard cells of the parent's neighbor at the same refinement level. In Fig. 3, the parent's neighbor is the coarse grid block on the right side of the refinement boundary. Since we are using the second approach to multigrid outlined in Section 2, we do not relax in the parent's neighbor block until we step down the multigrid hierarchy. Thus, the guard cell values assigned to a parent's neighbor by the Paramesh guard cell filling routine are not used by AMRMG.

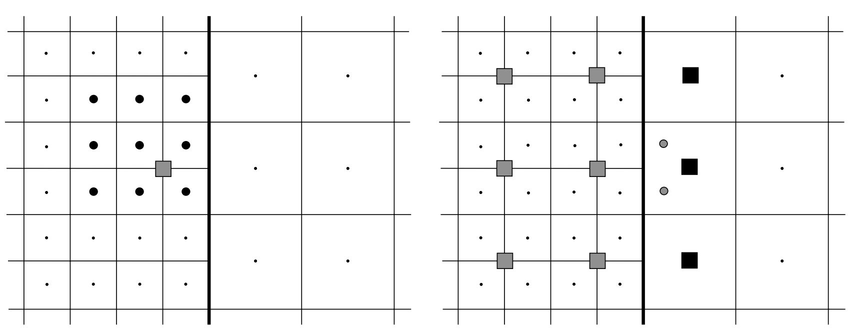

Fig. 3. The picture on the left shows the first step in guard cell filling, in which one of the parent grid cells (gray square) is filled using quadratic interpolation across nine interior fine grid cells (black circles). The other parent grid cells are filled using corresponding stencils of nine interior fine grid cells. The picture on the right shows the second step in which two fine grid guard cells (gray circles) are filled using quadratic interpolation across nine parent grid values (squares). These parent grid values include one layer of guard cells (black squares) obtained from the parent's neighbor on the right side of the interface, and two layers of interior cells (gray squares).

4. The FAS algorithm

AMRMG uses the full approximation storage (FAS) multigrid algorithm [25,34]. In this section, we provide a brief overview of FAS, and discuss the order of restriction and prolongation operators used for stepping down and up the multigrid hierarchy.

We are interested in the nonlinear elliptic differential equation $E(\cdot) = \rho$, where E is a (possibly) nonlinear elliptic operator. To avoid confusion with notation for exact and approximate solutions, we use a centered "dot" as a placeholder for the unknown function. Now consider a simple multigrid hierarchy consisting of two grids, a fine grid at level 2 and a coarse grid at level 1. The differential equation $E(\cdot) = \rho$, discretized on the finest level, becomes $E_2(\cdot) = \rho_2$. Here, the subscripts denote the multigrid level. What we seek is a solution of the difference equations $E_2(\cdot) = \rho_2$.

The basic V-cycle for the FAS algorithm consists of the following steps.

Step 1. Guess a trial solution $\tilde{\phi}_2$ (for example, $\tilde{\phi}_2 = 0$) and carry out some number of relaxation sweeps on the equation $E_2(\cdot) = \rho_2$ to obtain an approximate solution ϕ_2.

Step 2. Construct the coarse grid source

$$\rho_1 = \mathscr{R}(\rho_2 - E_2(\phi_2)) + E_1(\mathscr{R}\phi_2). \tag{3}$$

Here, $\mathscr{R}$ denotes the restriction of data from multigrid level 2 to level 1. Also, E_1 denotes the discretization of the elliptic operator E on the coarse grid 1. Loosely speaking, the term $-\mathscr{R}(E_2(\phi_2))$ removes the predominantly short wavelength part of the source that the fine grid has already captured in Step 1. The term $E_1(\mathscr{R}\phi_2)$ returns the long wavelength part of the source that was removed by subtracting $\mathscr{R}(E_2(\phi_2))$.

Step 3. Start with the trial solution $\tilde{\phi}_1 = \mathscr{R}\phi_2$ and carry out some number of relaxation sweeps on the equation $E_1(\cdot) = \rho_1$ to obtain an approximate solution ϕ_1. Alternatively, if possible, solve $E_1(\cdot) = \rho_1$ exactly for ϕ_1.

Step 4. Construct the trial solution

$$\tilde{\phi}_2 = \phi_2 + \mathscr{P}(-\mathscr{R}\phi_2 + \phi_1). \tag{4}$$

Here, $\mathscr{P}$ denotes the prolongation of data from multigrid level 1 to level 2. Loosely speaking, the term $\mathscr{P}(-\mathscr{R}\phi_2 + \phi_1)$ removes the long wavelength part of ϕ_2 and replaces it with ϕ_1, which contains primarily long wavelength information due to the construction of ρ_1.

Step 5. Start with the trial solution $\tilde{\phi}_2$ from Step 4 and carry out some number of relaxation sweeps on the equation $E_2(\cdot) = \rho_2$ to obtain an improved approximate solution ϕ_2.

For successive V-cycles the approximate solution ϕ_2 from Step 5 is used as the trial solution $\tilde{\phi}_2$ in Step 1. The FAS V-cycle can be generalized in an obvious way to any number of multigrid levels.

At the bottom of each V-cycle Step 3 instructs us to find an exact or approximate solution of the equation $E_1(\cdot) = \rho_1$. The subscript '1' denotes the coarsest level in the multigrid hierarchy. It is important for the performance of any multigrid code to solve this equation accurately. Solving the level 1 equation can be a potential bottleneck for our code because Paramesh places the data for each block on a single processor. When the algorithm is at the bottom of a V-cycle, only a single processor is active.

The simplest strategy for solving the level 1 equation $E_1(\cdot) = \rho_1$ is to carry out relaxation sweeps, just as we do for the higher multigrid levels. Typically our coarsest multigrid level is a single data block with 8^3 interior grid points. We find that with Robin boundary conditions it typically takes about one hundred relaxation sweeps to solve the level 1 equation to sufficiently high accuracy. With fewer sweeps at this level the code can require more V-cycles to solve the elliptic problem.

The concern is that the solution of the level 1 equation, requiring ~100 sweeps with only one processor active, can dominate the run time for the code. However, it turns out that the run time for AMRMG is dominated by communications calls made in Paramesh. Currently AMRMG uses version 3.0 of Paramesh, which is one of the first versions of Paramesh to run under MPI. More recent versions are better optimized, but we have not switched to the latest version of Paramesh because of the special modifications that AMRMG requires.

One way that we have improved the performance of our code is to bypass the communications calls made by Paramesh when solving the level 1 equation. The data for the level 1 equation always resides on block #1 on processor #1, so no communication among processors is needed. We have bypassed the Paramesh guard cell filling routine by writing a routine that directly fills the guard cells of this block using the outer boundary conditions.

We are primarily interested in solving elliptic problems that are semi-linear, that is, problems in which the second order derivative terms are linear in the unknown field. For these problems the equation to be solved at the bottom of each V-cycle takes the form $\Delta_1\phi_1 = \rho_1$, where Δ_1 is the Laplacian operator (not necessarily on flat space with Cartesian coordinates) discretized on multigrid level 1. In these cases we have an alternative to relaxation, namely, direct matrix inversion: $\phi_1 = (\Delta_1)^{-1}\rho_1$. We have implemented matrix inversion for level 1 using the direct Gaussian elimination routine from the LAPACK libraries [35]. With Robin boundary conditions, we must solve the level 1 equation for ϕ values in the guard cells as well as interior cells. With 8^3 interior grid points and one layer of guard cells, we have 10^3 values to determine at the bottom of each V-cycle. Therefore the matrix to be inverted has dimensions 1000×1000. Our tests show that it takes longer (by a factor of ~10) to solve the level 1 equation by direct matrix inversion than by relaxation, assuming the Paramesh communications calls have been bypassed. However, in either case the time required to solve the level 1 equation is a small fraction of the overall runtime for the code. Thus, we prefer to use the direct matrix inversion whenever possible because, with matrix inversion, the accuracy of the level 1 solution is insured.

5. Restriction and prolongation

As described in Section 3, guard cells are filled with a combination of restriction and prolongation operations. The operators used for guard cell filling are third order accurate, and we denote these by ${}^{(3)}\mathscr{R}$ for restriction and ${}^{(3)}\mathscr{P}$ for prolongation. What order restriction and prolongation operators do we use for stepping down and up the multigrid hierarchy? The answer is that we use a combination of second, third, and fourth order operators.

The restriction operators in Paramesh are always defined in such a way that only interior cells from the child blocks are used to fill the interior cells of a parent block. The fine grid stencil is positioned to keep the coarse grid point as close as possible to the center of the stencil. For the case of second order restriction, ${}^{(2)}\mathscr{R}$, the coarse grid point lies at the center of the stencil and gets its value from a succession of linear interpolations in each dimension. The case of third order restriction, ${}^{(3)}\mathscr{R}$, is depicted in the left panel of Fig. 3 and is described in Section 3.

The second, third, and fourth order prolongation operators in Paramesh use a succession of (respectively) linear, quadratic, and cubic interpolations in each dimension to fill the fine grid cells. The prolongation operators use both interior cells and guard cells from a parent block to fill both interior and guard cells of child blocks. The right panel of Fig. 3 shows the stencil used by the third order prolongation operator, ${}^{(3)}\mathscr{P}$, to fill fine grid guard cells on the right side of a fine grid block. This same stencil is used to fill the first layer of interior cells (the layer of interior fine grid cells adjacent to the block boundary). For the second and third layer of interior fine grid cells, the stencil is shifted to the left by one coarse grid point. This pattern of stencil shifting continues across the right half of the fine grid block until the midpoint of the block is reached. The stencils used for the left half of the fine grid block are the mirror images of those used for the right half.

The version of Paramesh currently used by AMRMG allows for second and third order restriction, and (in principle) arbitrary order prolongation. We have carried out many numerical tests to help us choose among different combinations of restriction and prolongation operators for stepping down and up the V-cycles. For all of these tests we used third order restriction and third order prolongation to fill the fine grid guard cells, as described in Section 3. In our tests we did not consider prolongation orders higher than four.

As we have presented it, the FAS algorithm uses two restriction operators in Step 2, one restriction operator in Step 3, and one restriction operator in Step 4. It uses one prolongation operator in Step 4. One could consider distributing the (assumed linear) restriction operator through the first term in Eq. (3) and treating the operators independently. Likewise, one could consider distributing prolongation operator through the second term in Eq. (4) and treating the operators independently. We have not considered the consequences of splitting these terms. Moreover, AMRMG is written such that the calculation $\mathscr{R}\phi_2$ from Step 2 is used as the trial solution $\tilde{\phi}_1 = \mathscr{R}\phi_2$ for Step 3. Thus, we have not tested the consequences of treating these restriction operators independently. Note that the restriction of the fine grid solution, $\mathscr{R}\phi_2$, appears in Step 4 as well as Steps 2 and 3. Our tests show that the order of the restriction operator in Step 4 must agree with the order used for $\mathscr{R}\phi_2$ in Steps 2 and 3. If not, the algorithm will often fail to converge in the sense that the residual (defined below) will not decrease with successive V-cycles.

The options that remain for restriction and prolongation operators can be expressed by rewriting Eqs. (3) and (4):

$$\rho_1 = {}^{(b)}\mathscr{R}(\rho_2 - E_2(\phi_2)) + E_1({}^{(a)}\mathscr{R}\phi_2), \tag{5a}$$

$$\tilde{\phi}_2 = \phi_2 + {}^{(c)}\mathscr{P}(-{}^{(a)}\mathscr{R}\phi_2 + \phi_1). \tag{5b}$$

The letters a, b, and c represent the orders of restriction and prolongation operators that appear in stepping down and up the V-cycles.

We want to find values for a, b, and c that will give the best performance. In judging the performance of our code we are looking to see how quickly the residual decreases with successive V-cycles for a fixed non-uniform mesh. The residual at each point in the computational domain is defined by res = $\rho_n - E_n(\phi_n)$, where n is the highest refinement level at that point and ϕ_n is the approximate solution. The norm of the residual is computed as

$$\langle \mathrm{res} \rangle = \sqrt{\frac{1}{N}\left(\sum \mathrm{res}^2\right)}. \tag{6}$$

The sum extends over the grid points that cover the computational domain at the highest refinement level. (In Fig. 2 these would be the interior points of blocks 4A–5A–5B–5C–5D–4D–3C–4E–4F.) The number N is the total number of such grid points. The norm $\langle \mathrm{res} \rangle$ defined above is similar to the usual L_2 norm, but lacks a factor of the cell volume in the "measure" of the sum. That is, the usual L_2 norm would be written as $\sqrt{(\sum v\mathrm{res}^2)/V}$, where v is the volume of each grid cell and V is the total volume of the computational domain. By omitting the factors of cell volume, the norm $\langle \mathrm{res} \rangle$ gives equal weighting to the residuals in each grid cell, regardless of resolution.

To be specific, we will quote the results for the simple test problem $\Delta\phi = \rho$, where Δ is the flat space Laplacian in Cartesian coordinates. We use the source $\rho = (6 - 9r^3)\exp(-r^3)$ with $r = \sqrt{x^2 + y^2 + z^2}$. At the boundaries we use the Robin condition $\frac{\partial}{\partial r}[r(\phi - 1)] = 0$. The analytic solution to this problem is $\phi = 1 + (1 - \exp(-r^3))/r$. The numerical solution is computed with a fixed three-level "box-in-box" grid structure. The highest resolution region has cell size $\Delta x = 0.125$ and covers a cubical domain with x, y, and z ranging from -2 to 2. The medium resolution region has cell size $\Delta x = 0.25$. It covers the domain from

-4 to 4 that is exterior to the high resolution region. The low resolution region has cell size $\Delta x = 0.5$. It covers the domain from -8 to 8 that is exterior to the medium resolution region. It has been our experience that the generic behavior of AMRMG is fairly well represented by this simple test case.

Our first observation is that with $a = 2$ the residual gets ''stuck'' after a few V-cycles. The norm $\langle \text{res} \rangle$ drops to about 10^{-4}, but no further. This happens regardless of the values chosen for b and c. In the limit of high resolution the truncation error is less than the residual, and the code fails to show second order convergence of the solution. Thus, we can eliminate the cases in which $a = 2$ and focus on $a = 3$.

For $a = 3$ the norm of the residual decreases with successive V-cycles to values well below the truncation error. Table 1 shows the average change in the common logarithm of $\langle \text{res} \rangle$ for each V-cycle, as a function of the number of relaxation sweeps at each multigrid level. (This excludes the first multigrid level, at the bottom of each V-cycle, where we compute the exact solution using matrix inversion.) The best performance is obtained with $(a, b, c) = (3, 2, 4)$. Note that the norm of the residual becomes insensitive to the number of relaxation sweeps as the number of sweeps increases beyond four or five. This is because, as observed in Section 2, the higher multigrid levels that have lower resolution neighbors can only receive updated boundary information once each V-cycle. It does not help to continue relaxation sweeps when the boundary information is ''old'' and needs to be updated. We typically use four relaxation sweeps, with red-black Gauss–Seidel ordering [34].

The results of our testing lead to the following formulas for Steps 2 and 4 of the FAS algorithm:

$$\rho_1 = {}^{(2)}\mathscr{R}(\rho_2 - E_2(\phi_2)) + E_1({}^{(3)}\mathscr{R}\phi_2), \tag{7a}$$

$$\tilde{\phi}_2 = \phi_2 + {}^{(4)}\mathscr{P}(-{}^{(3)}\mathscr{R}\phi_2 + \phi_1). \tag{7b}$$

In Step 3, we use the trial solution $\tilde{\phi}_1 = {}^{(3)}\mathscr{R}\phi_2$. Recall that we have not tested the algorithm with order of restriction greater than 3, or with order of prolongation greater than 4.

Conventional wisdom for determining the orders of restriction and prolongation used for multigrid transfer operations is that the following should be satisfied:

$$\mathscr{O}_{\mathscr{R}} + \mathscr{O}_{\mathscr{P}} > \mathscr{O}_{\mathscr{D}}. \tag{8}$$

Here, $\mathscr{O}_{\mathscr{R}}$, $\mathscr{O}_{\mathscr{P}}$, and $\mathscr{O}_{\mathscr{D}}$ are the orders of restriction, prolongation, and the differential operator, respectively [36]. For a uniform grid, AMRMG acts as a typical FAS multigrid solver and we achieve acceptable convergence rates as long as the transfer operators satisfy Eq. (8). With a nonuniform grid, the restriction operator denoted ${}^{(a)}\mathscr{R}$ in Eqs. (5) must be third or higher order, at least in the vicinity of mesh refinement boundaries, for the code to achieve both second order accuracy and optimal convergence rates. This is not surprising since, with the MLAT approach, data that is restricted from a high resolution region (for example, data restricted from block 5D to block 4C in Fig. 1) can serve as boundary data for relaxation in a coarse grid region (block 4D in Fig. 1). With restriction order less than 3, such boundary data yield truncation errors greater than $\mathscr{O}(\Delta x)$ when they appear in a discrete second derivative. One of our goals

Table 1
Average change in $\log(\langle \text{res} \rangle)$ per V-cycle

Number of sweeps	Order of restriction and prolongation operators (a, b, c)					
	(3, 2, 2)	(3, 2, 3)	(3, 2, 4)	(3, 3, 2)	(3, 3, 3)	(3, 3, 4)
1	−0.00	−0.11	−0.16	−0.11	−0.13	−0.33
2	−0.32	−0.42	−0.50	−0.33	−0.46	−0.56
3	−0.56	−0.75	−0.77	−0.43	−0.56	−0.69
4	−0.77	−0.88	−1.13	−0.50	−0.63	−0.74
5	−0.85	−1.08	−1.21	−0.56	−0.68	−0.79
6	−0.87	−1.15	−1.23	−0.62	−0.74	−0.83

for AMRMG is to achieve second order accuracy and good convergence rates with minimal modification of the existing Paramesh framework. For this reason we have not explored the possibility of using modified finite difference stencils or modified transfer operators in the vicinity of mesh refinement boundaries.

6. Truncation error and grid control

AMRMG adapts the grid structure to the problem at hand in an attempt to keep the local truncation error under control. The local truncation error is defined across the computational domain on the grid that consists of the highest resolution blocks. In Fig. 2 these would be blocks 4A–5A–5B–5C–5D–4D–3C–4E–4F. Let us refer to this non–uniform grid as grid h. Then the local truncation error is defined by [25]

$$\tau_h = E_h(\phi|_h) - (E(\phi))|_h, \tag{9}$$

where ϕ is the exact solution of the continuum equation $E(\phi) = \rho$, and $\phi|_h$ is the projection of ϕ onto grid h. The discretization of the operator E on grid h is denoted by E_h. In a similar manner, we define the local truncation error τ_H on a grid H that is constructed from the parents of grid h blocks. Grid H covers the computational domain with half the resolution of grid h. The difference between the truncation errors on grids H and h is

$$\tau_H - \mathscr{R}\tau_h = E_H(\phi|_H) - (E(\phi))|_H - \mathscr{R}E_h(\phi|_h) + \mathscr{R}(E(\phi))|_h, \tag{10}$$

where $\mathscr{R}$ is a linear operator that restricts data from h to H. Let us assume that $\mathscr{R}$ is third order accurate in the grid spacing. Then the second and fourth terms in Eq. (10) cancel to third order and, to this same order of accuracy, we find $\tau_H - \mathscr{R}\tau_h \approx E_H(\phi|_H) - \mathscr{R}E_h(\phi|_h)$.

The relative local truncation error is defined on grid H by [25]

$$\tau_h^H = E_H(\mathscr{R}\tilde{\phi}_h) - \mathscr{R}(E_h(\tilde{\phi}_h)), \tag{11}$$

where $\tilde{\phi}_h$ is the approximate numerical solution from grid h. Again we assume that the restriction operator is accurate to third order. Since the approximate solution coincides with the exact solution to leading order, $\tilde{\phi}_h \approx \phi|_h$, we see from Eqs. (10) and (11) that to leading order in the grid spacing the relative truncation error is related to the local truncation errors by $\tau_h^H \approx \tau_H - \mathscr{R}\tau_h$. Since we use second order differencing for our elliptic problems the truncation errors are proportional to the square of the grid spacing. Then $\tau_H \approx 4\mathscr{R}\tau_h$ and the relative truncation error is given by $\tau_h^H \approx 3\mathscr{R}\tau_h$. This relation can be prolonged to the finest grid h, giving $\mathscr{P}\tau_h^H \approx 3\mathscr{P}\mathscr{R}\tau_h$. The prolongation operator $\mathscr{P}$, like the restriction operator $\mathscr{R}$, is assumed to be third order accurate in the grid spacing. Then to third order accuracy $\mathscr{P}\mathscr{R}$ is the identity operator on h, and we have

$$\tau_h \approx \frac{1}{3}\mathscr{P}\tau_h^H. \tag{12}$$

Together, Eqs. (11) and (12) give

$$\tau_h \approx \frac{1}{3}\left({}^{(3)}\mathscr{P}E_H({}^{(3)}\mathscr{R}\tilde{\phi}_h) - E_h(\tilde{\phi}_h)\right). \tag{13}$$

In AMRMG, we use this approximation of the local truncation error to monitor the errors and control the grid structure. Note that since the truncation error τ_h is proportional to the square of the grid spacing, the result (13) is valid to leading order only if third (or higher) order restriction and prolongation operators are used.

For the test problem described in Section 5, it is straightforward to calculate the analytic truncation error. Fig. 4 shows a comparison of the analytic truncation error with the computed approximation to

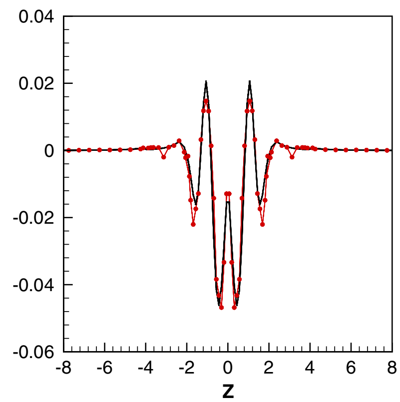

Fig. 4. Analytic truncation error and calculated truncation error along the z–axis.

the truncation error (13) for that test problem. The analytic truncation error is shown as a thick solid line, while the calculated truncation error is a thin line with filled circles.

In practice, we start the solution process by specifying a rather coarse grid structure, sometimes uniform sometimes not. The code V-cycles until the norm of the residual across grid h is less than the norm of the local truncation error across grid h. We write this as $\langle\text{res}\rangle|_h < \langle\tau_h\rangle|_h$, where the norm of the truncation error is defined in the same way as the norm of the residual, Eq. (6). The code usually requires two or three V-cycles to meet this criterion. We then compute the norm of the truncation error $\langle\tau_h\rangle|_b$ for each block of grid h. Any block whose norm is greater than some threshold value, $\langle\tau_h\rangle|_b > \tau_{\max}$, is flagged for refinement. Paramesh rebuilds the grid structure and redistributes the data across processors. To obtain a trial solution in the newly formed blocks we prolong the solution from the parent blocks. The code then carries out V-cycles on this new grid structure with its new highest-resolution grid h. The entire process repeats until all blocks satisfy $\langle\tau_h\rangle|_b \leqslant \tau_{\max}$ and no blocks are flagged for refinement. At this point the code continues to V-cycle until two conditions are satisfied: (1) the norm of the residual in each block is less than the norm of the truncation error, $\langle\text{res}\rangle|_b < \langle\tau_h\rangle|_b$; and (2) the norm of the residual across the entire grid is less than some threshold value, $\langle\text{res}\rangle|_h < \text{res}_{\max}$. If only the first condition is desired, we simply set $res_{\max}$ to a very large value.

We have tested the code using second, third, and fourth order prolongation operators for the calculation of trial solutions in newly formed blocks. We find that none of these operators is consistently better than the others. The entire adaptive mesh, multigrid algorithm is not very sensitive to the order of prolongation used in this step. We typically use the fourth order operator ${}^{(4)}\mathscr{P}$.

The grid control scheme used by AMRMG works well. It insures that the truncation error in each block across the computational domain is uniformly low, less than $\tau_{\max}$, and that the errors coming from the residuals in each block are less than the truncation errors. The value chosen for $\tau_{\max}$ depends on the problem being solved and the desired degree of accuracy.

7. Code tests

In this section, we present code tests to demonstrate second order convergence and the computational advantages of AMR. Fig. 5 shows a comparison of errors for the test case described in Section 5.

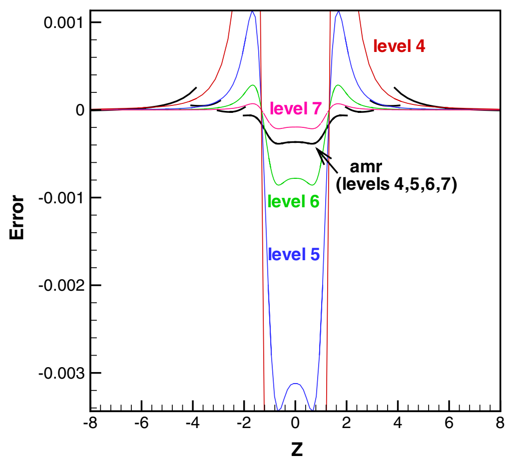

Fig. 5. Errors along the z–axis for AMR grid and uniform grids of various resolutions.

The analytical errors on the z-axis are plotted for an AMR grid and for uniform grids with resolutions at levels 4, 5, 6 and 7. (A grid with resolution level X is created by refining a single block $X - 1$ times. See Fig. 1.) For the AMR run we start with a uniform level 3 grid and set the refinement criterion for a maximum truncation error of $\tau_{\max} = 0.001$ in each block. We also limited the highest resolution to level 7, so the truncation errors in some of the level 7 blocks reached as high as 0.007. The cell size for resolution level 4 is $\Delta x = 0.25$, and the cell size for resolution level 7 is $\Delta x = 0.03125$. The grid structure chosen by AMRMG for the AMR run is shown in Fig. 6.

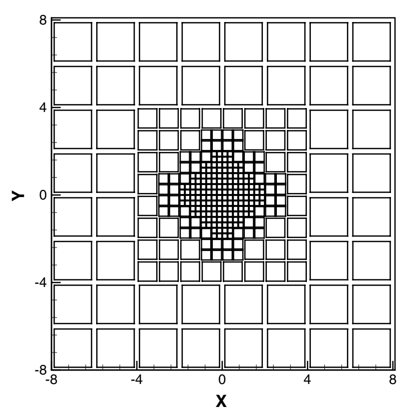

Fig. 6. AMR grid structure in the x-y plane. Each square corresponds to a block of data containing 8^3 computational cells.

Observe that the errors in each region of the AMR grid are comparable to the errors obtained with a uniform grid of the same resolution. For example, the errors in the level 7 region of the AMR solution (the region surrounding the origin) are slightly larger than the errors obtained from the uniform level 7 solution, and smaller than the errors obtained from the uniform level 6 solution. We also note that the savings in memory with AMR is profound; the AMR grid solution can be calculated on a single processor, while the level 7 uniform grid solution required 64 processors to handle the memory requirements.

As a test for second order convergence, we consider the initial data for a single black hole. Valid initial data must satisfy the constraint equations of general relativity. For a vacuum spacetime, the Hamiltonian and momentum constraints are given, respectively, by

$$R + K^2 - K_{ij}K^{ij} = 0 \tag{14}$$

and

$$D_j(K^{ij} - g^{ij}K) = 0, \tag{15}$$

where g_{ij} is the physical metric and g^{ij} is its inverse. Also, R is the scalar curvature, K_{ij} is the extrinsic curvature with trace $K = K^i_i$, and D_j is the covariant derivative associated with the spatial metric. These initial value equations must be rewritten as a well-posed elliptic boundary value problem. The standard techniques for rewriting the constraint equations are based on the York–Lichnerowicz conformal decomposition [13]. Following this approach, we assume that the physical metric g_{ij} is conformally related to a background metric $\tilde{g}_{ij}$,

$$g_{ij} = \psi^4 \tilde{g}_{ij}, \tag{16}$$

where ψ^4 is the conformal factor. The physical extrinsic curvature is written as

$$K_{ij} = \psi^{-2}\tilde{K}_{ij}. \tag{17}$$

In terms of these conformal variables, the Hamiltonian and momentum constraints read

$$8\tilde{\nabla}^2\psi - \psi\tilde{R} + \psi^{-7}\left(\tilde{K}_{ij}\tilde{K}^{ij} - \tilde{K}^2\right) = 0, \tag{18a}$$

$$\tilde{\nabla}_i(\tilde{K}^{ij} - \tilde{g}^{ij}\tilde{K}) = -4\psi^{-1}\tilde{K}\tilde{\nabla}^i\psi, \tag{18b}$$

where $\tilde{\nabla}_i$ and $\tilde{R}$ are the covariant derivative and scalar curvature associated with the background metric $\tilde{g}_{ij}$.

The "puncture method" [37] is a way of specifying black hole initial data on $\mathbb{R}_3$. The background metric is chosen to be flat. The momentum constraint (18b) is solved analytically by [38]

$$\tilde{K}^{ij} = \frac{3}{2r^2}(P^i n^j + P^j n^i - (\tilde{g}^{ij} - n^i n^j)P^k n_k), \tag{19}$$

where P^i is the momentum of the black hole and n^i is the radial normal vector (in the flat background with Cartesian coordinates). Note that $\tilde{K}_{ij}$ is traceless, $\tilde{K} = 0$. This expression (19) for $\tilde{K}_{ij}$ can be generalized to include an arbitrary number of black holes with spin and momentum, but for simplicity we will use a single black hole with no spin for our test case. To complete the specification of initial data, we must solve the Hamiltonian constraint (18a) for the conformal factor ψ. With the puncture method, the solution ψ is split into a known singular term and a nonsingular term u:

$$\psi = u + \frac{m}{2|\vec{r}|}. \tag{20}$$

Here, m is the "bare mass" of the black hole and $|\vec{r}|$ is the coordinate distance from the origin. With the puncture method splitting of the conformal factor, the Hamiltonian constraint becomes

$$\nabla^2 u + \beta(1+\alpha u)^{-7} = 0, \tag{21}$$

where

$$\beta = \frac{1}{8}\alpha^7 \tilde{K}^{ij}\tilde{K}_{ij} \quad \text{and} \quad \alpha = \frac{2|\vec{r}|}{m}. \tag{22}$$

Eq. (21) is solved for the nonsingular function u on $\mathbb{R}_3$ with Robin boundary conditions $\frac{\partial}{\partial r}[r(\phi - 1)] = 0$.

For these tests we use a fixed mesh refinement (FMR) grid with an "X plus 3" ($Xp3$) structure. The terminology $Xp3$ means that the grid is composed of 4 refinement regions, with the coarsest part of the grid at level X and the finest part of the grid at level $X + 3$. The different levels are nested in a "box-in-box" fashion. The finest level, with resolution $X + 3$, extends from -2 to 2 in each coordinate direction. The level with resolution $X + 2$ covers the domain between -4 and 4, excluding the finest level. The level with resolution $X + 1$ covers the domain between -8 and 8 excluding the finer levels. The coarsest level, with resolution X, covers the domain between -16 and 16 excluding the finer levels.

For our test case we have chosen $m = 1$ and $P^i = (0, 0, 1)$. We solve Eq. (21) using the series of FMR grids $3p3$, $4p3$, $5p3$, $6p3$, and $7p3$. Each successive FMR grid has the same boundaries and double the resolution of the previous grid. Fig. 7 shows a contour plot of the solution u in the y–z plane, obtained with the $6p3$ grid.

Figs. 8 and 9 show the results of a three–point convergence test for data along the z axis. This data passes through the puncture (at the origin) where the solution u and its derivatives are changing most rapidly. The three–point convergence test is obtained by plotting the difference between solutions on successive FMR grids, multiplied by an appropriate power of 4. The top (red) curve shown in Figs. 8 and 9 is the difference between the solution u obtained on the $4p3$ grid and the solution obtained on the $3p3$ grid. The next curve is the difference between solutions on FMR grids $5p3$ and $4p3$, multiplied by 4. The third curve is the difference between solutions on FMR grids $6p3$ and $5p3$, multiplied by 16. Finally, the curve that has the most negative value at the origin in Fig. 9 is the difference between solutions on the FMR grids $7p3$ and $6p3$, multiplied by 64. One can see that the curves in Figs. 8 and 9 overlay one another in the limit of high resolution. This shows that the errors in AMRMG are second order in the grid spacing.

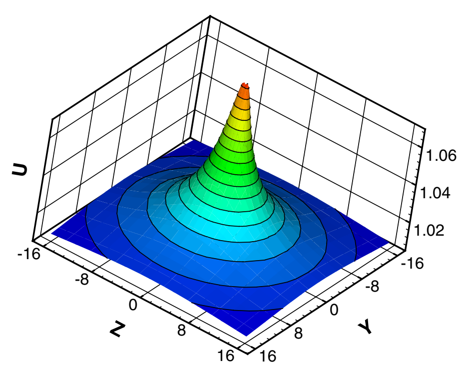

Fig. 7. Contour plot of the non-singular part of solution u.

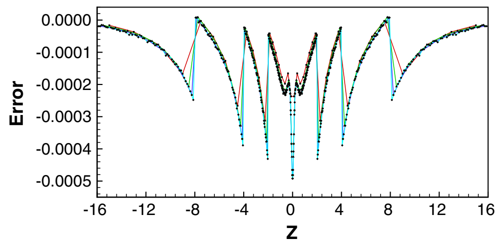

Fig. 8. Three point convergence test of puncture data.

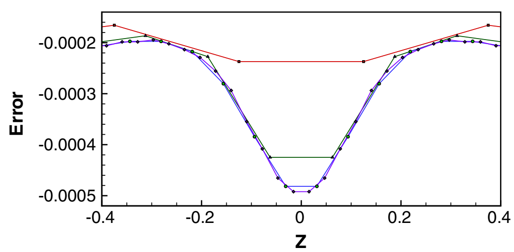

Fig. 9. Three point convergence test of puncture data close to puncture.

Acknowledgements

We thank the numerical relativity group at NASA Goddard Space Flight Center, and especially Dae-Il Choi, for their help and support. We would also like to thank Kevin Olson and Peter MacNeice for their help with Paramesh. This work was supported by NASA Space Sciences Grant ATP02-0043-0056. Computations were carried out on the North Carolina State University IBM Blade Center Linux Cluster. The Paramesh software used in this work was developed at NASA Goddard Space Flight Center under the HPCC and ESTO/CT projects.

References

[1] M.W. Choptuik, Phys. Rev. Lett. 70 (1993) 9.
[2] B. Brügmann, Phys. Rev. D 54 (1996) 7361, gr-qc/9608050.
[3] B. Brügmann, Int. J. Mod. Phys. D 8 (1999) 85, gr-qc/9708035.
[4] P. Papadopoulos, E. Seidel, L. Wild, Phys. Rev. D 58 (1998) 084002, gr-qc/9802069.
[5] P. Diener, N. Jansen, A. Khokhlov, I. Novikov, Class. Quant. Grav. 17 (2000) 435, gr-qc/9905079.
[6] K.C.B. New, D.-I. Choi, J.M. Centrella, P. MacNeice, M. Huq, K. Olson, Phys. Rev. D 62 (2000) 084039, gr-qc/0007045.
[7] M.W. Choptuik, E.W. Hirschmann, S.L. Liebling, F. Pretorius, Phys. Rev. D 68 (2003) 044007, gr-qc/0305003.
[8] D.-I. Choi, J.D. Brown, B. Imbiriba, J. Centrella, P. MacNeice, J. Comput. Phys. 193 (2004) 398, physics/0307036.
[9] E. Schnetter, S.H. Hawley, I. Hawke, Class. Quant. Grav. 21 (2004) 1465, gr-qc/0310042.
[10] B. Brügmann, W. Tichy, N. Jansen, Phys. Rev. Lett. 92 (2004) 211101, gr-qc/0312112.
[11] B. Imbiriba, J. Baker, D.-I. Choi, J. Centrella, D.R. Fiske, J.D. Brown, J.R. van Meter, K. Olson, Phys. Rev. D 70 (2004) 124025, gr-qc/0403048.

[12] C. Misner, K. Thorne, J.A. Wheeler, Gravitation, W.H. Freeman, San Francisco, 1973.
[13] G.B. Cook, Living Rev. Rel. 3 (2000) 5, gr-qc/0007085.
[14] L.E. Kidder, M.A. Scheel, S.A. Teukolsky, Phys. Rev. D 64 (2001) 064017, gr-qc/0105031.
[15] L. Lindblom, M.A. Scheel, Phys. Rev. D 66 (2002) 084014, gr-qc/0206035.
[16] M.A. Scheel, L.E. Kidder, L. Lindblom, H.P. Pfeiffer, S.A. Teukolsky, Phys. Rev. D 66 (2002) 124005, gr-qc/0209115.
[17] L. Lindblom, et al., Phys. Rev. D69 (2004) 124025, gr-qc/0402027.
[18] M.W. Choptuik, E.W. Hirschmann, S.L. Liebling, F. Pretorius, Class. Quant. Grav. 20 (2003) 1857, gr-qc/0301006.
[19] M. Anderson, R.A. Matzner, (2003), gr-qc/0307055.
[20] R.A. Matzner, Phys. Rev. D71 (2005) 024011, gr-qc/0408003.
[21] R. Fedorenko, USSR Comp. Math. Math. Phys. 1 (1962) 1092.
[22] R. Fedorenko, USSR Comp. Math. Math. Phys. 4 (1964) 559.
[23] N.S. Bakhvalov, USSR Compt. Math. Math. Phys. 6 (1966) 861.
[24] A. Brandt, Math. Comput. 31 (1977) 333.
[25] A. Brandt, in: W. Hackbusch, U. Trottenberg (Eds.), Multigrid Methods, Springer–Verlag, Berlin, 1977.
[26] D. Martin, K. Cartwright, unpublished, 1996, Available from: <http://seesar.lbl.gov/anag/staff/martin/tar/AMR.ps>.
[27] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, Society for Industrial and Applied Mathematics, Philadelphia, 2000.
[28] P. MacNeice, K.M. Olson, C. Mobarry, R. deFainchtein, C. Packer, Comp. Phys. Comm. 126 (2000) 330.
[29] http://ct.gsfc.nasa.gov/paramesh/Users_manual/amr.html.
[30] J.D. Brown, L.L. Lowe, Phys. Rev. D70 (2004) 124014, gr-qc/0408089.
[31] S.F. McCormick, J. Thomas, Math. Comput. 46 (1986) 439.
[32] D. Bai, A. Brandt, J. Sci. Stat. Comput. 8 (1987) 109.
[33] G. Chesshire, W.D. Henshaw, J. Compt. Phys. 90 (1990) 1.
[34] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes, Cambridge University Press, Cambridge, 1992.
[35] http://www.netlib.org/lapack/index.html.
[36] S.A.U. Trottenberg, C.W. Oosterlee, Multigrid, Academic Press, London, 2001.
[37] S. Brandt, B. Brügmann, Phys. Rev. Lett. 78 (1997) 3606, gr-qc/9703066.
[38] J.M. Bowen, J. York, W. James, Phys. Rev. D21 (1980) 2047.

Chapter 5

Distorted Black Hole initial data using the puncture method

J. David Brown and Lisa L. Lowe
Distorted Black Hole Initial Data Using the Puncture Method
Physical Review D

PHYSICAL REVIEW D **70,** 124014 (2004)

Distorted black hole initial data using the puncture method

J. David Brown

Department of Physics, North Carolina State University, Raleigh, North Carolina 27695 USA

Lisa L. Lowe

Department of Physics, North Carolina State University, Raleigh, North Carolina 27695 USA

(Received 30 August 2004; published 10 December 2004)

We solve for single distorted black hole initial data using the puncture method, where the Hamiltonian constraint is written as an elliptic equation in $\mathbb{R}^3$ for the nonsingular part of the metric conformal factor. With this approach we can generate isometric and nonisometric black hole data. For the isometric case, our data are directly comparable to those obtained by Bernstein *et al.*, who impose isometry boundary conditions at the black hole throat. Our numerical simulations are performed using a parallel multigrid elliptic equation solver with adaptive mesh refinement. Mesh refinement allows us to use high resolution around the black hole while keeping the grid boundaries far away in the asymptotic region.

DOI: 10.1103/PhysRevD.70.124014 PACS numbers: 04.25.Dm, 04.30.Db, 04.70.Bw

I. INTRODUCTION

Distorted black holes are expected to be produced by astrophysical events such as asymmetrical gravitational collapse, black hole-black hole coalescence, black hole-neutron star coalescence, and possibly neutron star-neutron star coalescence. As ground based gravitational wave detectors such as LIGO, VIRGO, TAMA, and GEO increase their sensitivities, and the planned launch date for the space based detector LISA grows near, the need for researchers to develop a theoretical understanding of these systems becomes increasingly important. In support of that effort, we need to develop the techniques and tools to evolve a dynamic, distorted black hole and extract the emitted gravitational radiation as it settles to a quiescent state.

One of the difficulties encountered in the numerical treatment of single and multiple black hole systems is the presence of nontrivial topologies. In the past, researchers have addressed this problem by using isometry conditions at inner boundaries to represent black hole throats. Another approach to black hole evolution is excision, in which a region inside each apparent horizon is removed from the computational domain. Recently it has been shown that black holes can be treated in terms of fields on $\mathbb{R}^3$ by splitting the conformal factor for the spatial metric into singular and nonsingular terms. This so-called "puncture method" was applied to the Bowen-York [1] family of black hole initial data sets by Brandt and Brügmann [2], and used for evolution studies by Brügmann [3] and others (see, for example, Refs. [4,5]).

In this paper we make a modest extension of the puncture construction for initial data to include single distorted black holes. We reproduce and extend the results of Bernstein *et al.* [6–8], who constructed "black hole plus Brill wave" initial data sets using isometry conditions at the black hole throat. Whereas the black holes obtained by Bernstein *et al.* are isometric by construction, our distorted puncture black hole data sets are isometric only when the free parameters μ and m, defined in Sec. II, are equal to one another. For $\mu \neq m$, we obtain nonisometric, distorted black holes.

Another difficulty encountered in the numerical modeling of black holes is the wide discrepancy in length scales involved. The computational grid needs to be large enough to capture outgoing gravitational waves and to minimize boundary effects. The grid must also have high resolution in the interior to accurately resolve the strong gravitational fields of black holes. For a finite difference code, adaptive mesh refinement (AMR) is needed to satisfy both of these requirements: high resolution and a large grid. In this paper we introduce our AMR elliptic solver that allows us to solve accurately for puncture black hole data on very large grids. Evolution studies of these data sets are underway [9].

In Sec. II we set up the equations defining the initial value problem for a distorted puncture black hole. We show how the puncture data can be formulated to give isometric data sets, thus enabling a comparison with the results of Bernstein *et al.* In Sec. III we give a brief description of our AMR elliptic solver. In Sec. IV we present sample results both for isometric and nonisometric black hole data.

II. FORMULATION OF THE PROBLEM

Following Bernstein *et al.* [6–8] we write the line element for the physical metric g_{ij} as

$$ds^2 = \psi^4[e^{2q}(dr^2 + r^2 d\theta^2) + r^2 \sin^2\theta d\phi^2], \qquad (1)$$

where q is a specified function of the spatial coordinates and ψ is the conformal factor. In this paper we restrict ourselves to the expression for q used in Ref. [7], namely,

$$q(r, \theta, \phi) = 2Q_0 e^{-\eta^2} \sin^n\theta(1 + c\cos^2\phi), \qquad (2)$$

where $\eta = \ln(2r/\mu)$. In Eq. (2), Q_0, n, c and μ are adjustable constants that affect the size and type of distortion. We also restrict ourselves to data sets defined at a moment of time symmetry. Thus, the extrinsic curvature vanishes and the momentum constraints are trivially satisfied. The Hamiltonian constraint reduces to

$$\hat{\nabla}^2\psi - \frac{1}{8}\hat{R}\psi = 0, \qquad (3)$$

where $\hat{\nabla}^2$ and $\hat{R}$ are the Laplacian and scalar curvature of the conformal metric, defined by $\hat{g}_{ij} = \psi^{-4} g_{ij}$.

Solutions of Eq. (3) on the manifold $\mathbb{R}^3$ are Brill waves. The distorted black hole data sets of Bernstein *et al.* were obtained by solving Eq. (3) on the manifold $\mathbb{R} \times S^2$, with Robin conditions at the outer boundary [$\partial(r\psi - r)/\partial r = 0$ as $r \to \infty$] and isometry conditions at the inner boundary [$\partial(\sqrt{r}\psi)/\partial r = 0$ at $r = \mu/2$]. We will obtain solutions of Eq. (3) on $\mathbb{R} \times S^2$ by following the puncture construction of Ref. [2]. Thus, we write the conformal factor as $\psi = u + m/(2r)$ and insert this into the Hamiltonian constraint (3) to obtain

$$\hat{\nabla}^2 u - \frac{1}{8}\hat{R}u = \frac{m}{16r}\hat{R}. \qquad (4)$$

This equation is solved for a continuous solution u on $\mathbb{R}^3$ with Robin boundary conditions $\partial(ru - r)/\partial r = 0$ at $r \to \infty$. Note that the "bare mass" m appears as a new parameter in the construction.[1]

We have had no difficulty in obtaining numerical solutions to Eq. (4). However, from an analytical point of view, it is not immediately obvious to us whether or not Eq. (4) always admits solutions, and if so whether those solutions are unique. The problem of existence and uniqueness of solutions of Eq. (4) on $\mathbb{R}^3$ is similar to the problem of existence and uniqueness of solutions of Eq. (3) on $\mathbb{R}^3$. There are two notable differences. First is the appearance of a "source" term $m\hat{R}/(16r)$ on the right-hand side of Eq. (4). Note that with the choice (2) for the function q, the scalar curvature $\hat{R}$ goes to zero rapidly, $\hat{R} \sim [\ln(r)]^2 r^{\ln(\mu/r)-2-2\ln(2)}$ at $r \to 0$, so that $\hat{R}/r$ does not blow up at the puncture. The second difference between Eq. (4) and the familiar initial value Eq. (3) is that, for (4), we need not require the solutions to be positive. Rather, we would like to know if u is greater than $-m/(2r)$ so that the combination $\psi = u + m/(2r)$ is everywhere positive.

By construction the data sets of Bernstein *et al.* are isometric; that is, they are symmetric under reflections about the black hole throat $r = \mu/2$. More precisely, each data set is described by a metric tensor whose components g_{ij}, as functions of r, θ and ϕ, are unchanged by the coordinate transformation $r \to \bar{r} \equiv \mu^2/(4r)$. We can solve for reflection symmetric data sets using the puncture method as well, simply by setting the parameters μ and m equal to one another: $\mu = m$. To see this, we first note that the Hamiltonian constraint on $\mathbb{R} \times S^2$ can be written as (see, for example, Appendix D of Ref. [10])

$$\left(\tilde{\nabla}^2 - \frac{1}{8}\tilde{R}\right)(\sqrt{r}\psi) = 0, \qquad (5)$$

where $\tilde{\nabla}^2$ and $\tilde{R}$ are the Laplacian and scalar curvature for the metric $\tilde{g}_{ij} = \hat{g}_{ij}/r^2$. Let $\psi_1(r)$ denote a solution of Eq. (3), or equivalently Eq. (5), obtained by the puncture method. (For notational simplicity, we display only the r dependence in the solution ψ_1, and in ψ_2 below. In general these are functions of θ and ϕ as well.) This solution $\psi_1(r)$ has boundary behavior $\psi_1(r) \to 1$ as $r \to \infty$, and $\psi_1(r) \to m/(2r)$ as $r \to 0$. Now observe that the line element $ds^2 = \tilde{g}_{ij}dx^i dx^j$, with the function q chosen as in Eq. (2), is invariant under reflections $r \to \bar{r} = \mu^2/(4r)$. The scalar operator $\tilde{\nabla}^2 - \tilde{R}/8$ is invariant as well. By making the substitution $r \to \bar{r} = \mu^2/(4r)$ in Eq. (5) we see that $\psi_2(r)$, defined by $\sqrt{r}\psi_2(r) = \sqrt{\bar{r}}\psi_1(\bar{r})$ with $\bar{r} = \mu^2/(4r)$, is also a solution of Eqs. (5) and (3). This solution satisfies the boundary conditions $\psi_2(r) \to m/\mu$ as $r \to \infty$ and $\psi_2(r) \to \mu/(2r)$ as $r \to 0$. If we choose $\mu = m$, then the solutions $\psi_1(r)$ and $\psi_2(r)$ satisfy the same equation and obey the same boundary conditions. If we assume that solutions to the puncture Eq. (4) are unique, then the two solutions $\psi_1(r)$ and $\psi_2(r)$ must in fact be identical. From the equality $\psi_1(r) = \psi_2(r)$ we find

$$\sqrt{r}\psi_1(r) = [\sqrt{\bar{r}}\psi_1(\bar{r})]|_{\bar{r}=\mu^2/(4r)}. \qquad (6)$$

The line element (1) for this solution can be written as

$$ds^2 = [\sqrt{r}\psi_1(r)]^4[e^{2q}(dr^2/r^2 + d\theta^2) + \sin^2\theta d\phi^2]. \qquad (7)$$

Then the relation (6) is the condition for the physical metric (7) to be invariant under the reflection $r \to \bar{r} = \mu/(4r)$.

In Sec. IV we present results for the conformal factor ψ for isometric ($\mu = m$) and nonisometric ($\mu \neq m$) data sets. For isometric data, we can check the reflection symmetry by comparing the ADM mass computed at the two infinities. At the "outer" infinity ($r \to \infty$) the ADM mass is given by [11]

$$M_\infty = -\frac{1}{2\pi}\oint_\infty d\hat{S}^i\hat{\nabla}_i\psi. \qquad (8)$$

Numerically, the integral is computed at the outer bound-

[1]The parameters μ and m are dimensionful. Thus, our data sets can be described in terms of the dimensionless parameters c, Q_0, n, the dimensionless ratio μ/m, and the dimensionless coordinates x/m, y/m, and z/m.

ary of the computational domain. The ADM mass at the "inner" infinity ($r = 0$) is found by expressing the metric (1) in coordinates $\bar{r}$, θ, ϕ, where $\bar{r} = m^2/(4r)$. We find

$$ds^2 = \left(1 + \frac{mu}{2\bar{r}}\right)^4 [e^{2q}(d\bar{r}^2 + \bar{r}^2 d\theta^2) + \bar{r}^2 \sin^2\theta d\phi^2], \quad (9)$$

where q and u are functions of $r = m^2/(4\bar{r})$, θ, and ϕ. Provided q goes to zero sufficiently rapidly as $\bar{r} \to \infty$, the metric is asymptotically flat at $r = 0$ with ADM mass given by

$$M_0 = mu|_{r=0}. \quad (10)$$

The choice (2) for q vanishes as $r \to 0$ like $q \sim r^{\ln(\mu/r)}$, faster than any power of r.

III. NUMERICAL CODE

Our numerical code solves Eq. (4) using multigrid techniques with mesh refinement. We use the Paramesh package [12,13] to implement mesh refinement and parallelization. Paramesh divides the computational grid into blocks, with each block containing N zones. A block of data is refined by bisection—that is, the block is divided into eight "child" blocks (in three spatial dimensions) each containing N zones.

Our code carries out multigrid V-cycles using the Full Approximation Storage (FAS) algorithm [14] on nonuniform grid structures, with zone-centered data. It works with both fixed mesh refinement (FMR) and adaptive mesh refinement (AMR). Working with FMR, we specify a nonuniform grid by hand. Working with AMR, we generally start with a coarse, uniform grid. The code V-cycles until the norm of the residual is less than the norm of the relative truncation error in each block. Any block whose relative truncation error is above a specified tolerance is flagged for refinement. Paramesh then rebuilds the grid, the data is prolonged from the old grid to the new, and the V-cycle process begins again. Working in this mode takes the place of the Full Multigrid (nested iteration) Algorithm [14], where the solution on the final grid structure is reached by a succession of V-cycles of increasing peak resolution.

We will provide details of the AMR-multigrid algorithm in a later publication, along with numerous code tests [15].

IV. RESULTS

Bernstein *et al.* [6–8] produced nonaxisymmetric distorted black holes using isometry conditions at the black hole throat. We have reproduced several of the initial data sets from Ref. [7] using the puncture method, by setting $\mu = m$ as described in Sec. II. As a check for isometry, we have confirmed that the ADM mass at the puncture, M_0, agrees with the ADM mass at infinity, M_∞, to several significant digits.[2]

Two tests were performed to confirm the consistency of our ADM mass calculations. For the first test we used a sequence of grids with fixed interior resolution but increasing outer boundary limits. We started by solving for the initial data on a grid with boundaries (in each dimension) at -13 and 13, and having two additional box-in-box refinement levels ranging (in each dimension) from -6.5 to 6.5 and from -3.25 to 3.25. The resolution on the finest of the three levels was $\Delta x = 0.050\,78$. Other grids in the sequence were created by adding, one at a time, another fixed box-in-box refinement level while doubling the size of the computational domain. This test was intended primarily as a check on the sensitivity of the ADM mass at infinity to the location of the outer boundary. We found that the ADM mass M_∞ is unaffected to six digits and the ADM mass M_0 is unaffected to seven digits. For the second test we used a sequence of grids consisting of a fixed three-level box-in-box structure but increasing resolution throughout the computational domain. This test was intended primarily to check the convergence properties of the ADM masses M_∞ and M_0. The test showed that the ADM masses converge with second order accuracy.

As a specific example let us consider the data set discussed in Ref. [7] with $c = -2$, $Q_0 = -0.5$, $n = 4$, and $\mu = m = 2$. The ADM masses for this data at infinity and at the puncture are $M_\infty = 2.2021$ and $M_0 = 2.2027$. Figure 1 shows a contour plot of the nonsingular part u of the conformal factor in the $z = 0$ plane. The contour levels crossing the x-axis between -100 and zero are, respectively, $\{1.0015, 1.005, 1.005, 1.0015, 0.96\}$. The contour levels crossing the y-axis between -100 and zero are, respectively, $\{1.0015, 1.005, 1.01, 1.02, 1.1\}$. The inset in Fig. 1 shows the range in x and y from -13 to 13. In the inset figure, the contours crossing the x-axis between -13 to zero are $\{1.005, 1.0015, 0.96\}$ and the contours crossing the y-axis between -13 and zero are $\{1.01, 1.02, 1.1\}$. The function u forms two peaks along the y-axis with maximum values 2.03, and two valleys along the x-axis with minimum values 0.296. The value of u at the puncture is 1.10. The profiles of u along the three axes look similar to those shown in Fig. 2.

With AMR, we are able to push the boundaries out quite far while maintaining high resolution around the puncture. For the data described above, the computational grid extends from -100 to 100 and has refined itself to reach a maximum relative truncation error of 0.008. The lowest resolution region is equivalent to a 64^3 grid with

[2] A direct numerical comparison of our ADM masses with those of Bernstein *et al.* is not possible, since their ADM masses are presented graphically. A visual comparison our ADM mass data with theirs shows good agreement.

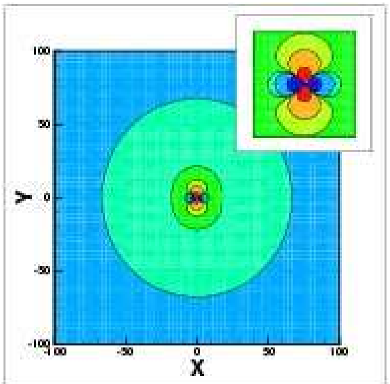

FIG. 1 (color online). Contour plot of u in the $z = 0$ plane for the case $c = -2$, $Q_0 = -0.5$, $n = 4$, $\mu = m = 2$. The full AMR grid extends from -100 to 100, while the inset shows the range -13 to 13.

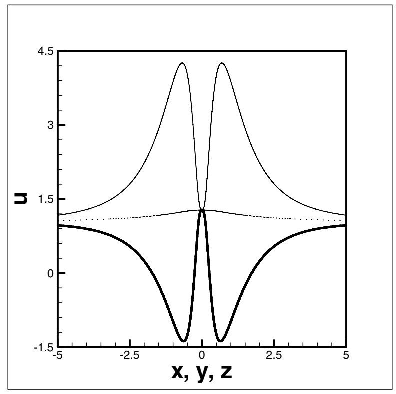

FIG. 2. The function u plotted along the three coordinate axes for a non-isometric distorted black hole with $c = -2$, $Q_0 = -0.5$, $n = 4$, $\mu = 2$, $m = 10$. The thin solid line (top curve) is u along the y-axis, the dotted line (middle curve) is u along the z-axis, and the thick solid line (bottom curve) is u along the x-axis.

$\Delta x = 3.125$. The highest resolution region is equivalent to a $262,144^3$ grid with $\Delta x = 0.000\,763$.

The puncture method allows us to define a new class of initial data: distorted black holes that are not isometric ($\mu \neq m$). Figure 2 shows such a data set with $c = -2$, $Q_0 = -0.5$, $n = 4$, $\mu = 2$, and $m = 10$. The results were computed on a grid extending from -100 to 100 in each dimension. The grid spacing for the lowest resolution region, adjacent to the grid boundaries, is $\Delta x = 3.125$. The grid spacing for the highest resolution region, surrounding the puncture, is $\Delta x = 0.012\,21$. The ADM mass at infinity is $M_\infty = 10.7$, while the ADM mass at the puncture is $M_0 = 12.8$. Figure 2 shows the behavior of the nonsingular part u of the conformal factor along the coordinate axes. The thin solid line plots u along the y-axis, the thick solid line plots u along the x-axis, and the dotted line plots u along the z-axis. The two peaks in u along the y-axis have maximum values 4.26, and the two valleys along the x-axis have minimum values -1.38. The value of u at the puncture is 1.28.

Note that for the nonisometric data described above, u takes on negative values in two small regions of radius ~ 1 at locations $\sim \pm 1$ along the x-axis. However, for this data set, the conformal factor $\psi = u + m/(2r)$ is everywhere positive. We have explored other data sets that contain regions with $u < 0$, and in each case we find that the conformal factor ψ is positive everywhere. For data with $c = -2$, $Q_0 = -0.5$, and $n = 4$, the most extreme data sets we have studied have ratios $\mu/m = 1/300$ and $\mu/m = 100$. For the case $\mu/m = 1/300$, the function u reaches a minimum value of ~ -120 but the conformal factor remains positive. For the case $\mu/m = 100$ the function u, and therefore also ψ, is always positive.

ACKNOWLEDGMENTS

We would like to thank the numerical relativity group at NASA Goddard Space Flight Center, and especially Dae-Il Choi, for their help and support. We would also like to thank James Isenberg, Kevin Olson, and Peter MacNeice for helpful discussions. This work was supported by NASA Space Sciences Grant No. ATP02-0043-0056. Computations were carried out on the North Carolina State University IBM Blade Center Linux Cluster. The PARAMESH software used in this work was developed at NASA Goddard Space Flight Center under the HPCC and ESTO/CT projects.

[1] J. M. Bowen and J.W. York, Jr., Phys. Rev. D **21**, 2047 (1980).

[2] S. Brandt and B. Brugmann, Phys. Rev. Lett. **78**, 3606 (1997).

[3] B. Brugmann, Int. J. Mod. Phys. D **8**, 85 (1999).
[4] M. Alcubierre *et al.*, Phys. Rev. D **67**, 084023 (2003).
[5] B. Imbiriba, J. Baker, D.-I. Choi, J. Centrella, D. R. Fiske, J. D. Brown, J. R. van Meter, and K. Olson, gr-qc/0403048.
[6] D. Bernstein, D. Hobill, E. Seidel, and L. Smarr, Phys. Rev. D **50**, 3760 (1994).
[7] S. Brandt, K. Camarda, E. Seidel, and R. Takahashi, Classical Quantum Gravity **20**, 1 (2003).
[8] S. R. Brandt and E. Seidel, Phys. Rev. D **54**, 1403 (1996).
[9] M. Koppitz *et al.* (to be published).
[10] R. M. Wald, *General Relativity* (University of Chicago Press, Chicago, 1984).
[11] N. Ó Murchadha and J.W. York, Phys. Rev. D **10**, 2345 (1974).
[12] http://ct.gsfc.nasa.gov/paramesh/Users_manual/amr.html.
[13] P. MacNeice, K. M. Olson, C. Mobarry, R. deFainchtein, and C. Packer, Comput. Phys. Commun. **126**, 330 (2000).
[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes* (Cambridge University Press, Cambridge, 1992).
[15] J. D. Brown and L. L. Lowe, gr-qc/041112.

Chapter 6

The ADM Evolution Equations

6.1 Introduction

The ADM evolution equations are a set of 12 time-dependent nonlinear partial differential equations for the dynamic variables g_{ij} and K_{ij}. Evolutions are performed by taking initial data that satisfies the constraint equations and evolving that data through time using the evolution equations. In addition to providing initial data for a solution, one must choose a gauge via the lapse and shift variables (α, β^i). Early attempts to solve these equations were plagued with inexplicable difficulties. Codes were invariably unstable and crashed shortly after beginning. Researchers experimented with various ways to remedy this, leading to extensive studies of numerical algorithms, boundary conditions, and gauge choices. Eventually it was understood that the equations themselves were inherently unstable and would have to be modified in order to produce stable, accurate, long running simulations. Several adaptations of the ADM formalism have been proposed, each with different stability properties usually depending on choice of gauge conditions. One of these approaches, BSSN[1], is being used to perform successful simulations of binary black hole mergers [25, 26, 27].

[1] Baumgarte–Shapiro–Shibata–Nakamura

6.2 Equations

The ADM evolution equations in their traditional form are

$$\partial_\perp g_{ij} = -2\alpha K_{ij} \tag{6.1}$$

$$\partial_\perp K_{ij} = \alpha K K_{ij} - 2\alpha K_{ik}K^k_j + \alpha R_{ij} - D_i D_j \alpha. \tag{6.2}$$

The mathematical nature of the evolution equations (6.1) and (6.2) are not well defined. Most equations can be classified as elliptic, parabolic, or hyperbolic[2]. For example, the initial value equations are elliptic. The evolution equations are classified as hyperbolic, but they are what is known as 'weakly hyperbolic' [46]. Ideally, we would like to evolve a system of equations that is strongly hyperbolic. Strongly hyperbolic (and in particular strongly hyperbolic of the symmetric type) systems can indicate well-posedness for a Cauchy problem [47][3]. Many modifications on the traditional ADM formalism are in an effort to recast the equations into a strongly hyperbolic form(e.g., BSSN[4],NOR [47],AA [51],EC [52] and others).

Numerical evolutions monitor the constraint equations as a test for accuracy and stability. On an initial time slice, the constraints are solved for appropriate data. Solutions that satisfy the constraints initially will theoretically satisfy the constraints for all time. In practice, however, there will always be small errors in the initial data, and there will always be small errors in the numerical computations during the evolution. This may lead one to concentrate on providing the most accurate initial data, using the finest resolution, coding the most sophisticated numerical integration scheme, and crafting the most realistic boundary conditions. While all these are fine aspirations, it turns out that the real problem is more fundamental; instability is inherent in the very form of the equations. The traditional ADM evolution equations admit what are known as 'constraint violating modes'. This means that the slightest

[2]For a second order linear partial differential equation $L(u) = Au_{tt} + Bu_{tx} + Cu_{xx} + Du_t + Eu_x + Fu = G$ where $u = u(x,t)$, $A = A(x,t)$, etc., an equation is hyperbolic if the determinant $B^2 - 4AC > 0$, elliptic if $B^2 - 4AC < 0$, and parabolic if $B^2 - 4AC = 0$.

[3]Well-Posed: A problem where a solution exists, is unique, and depends continuously on the data.

[4]While it was thought that BSSN was successful because of its inherent hyperbolicity [49], it has also been proposed that the success is due to being inherently constrained [50] [48].

deviation from the initial constrained system will not only cause the error in the constraint equations to grow but to grow exponentially.

There are infinite solutions to any evolution equation, as there are an infinite number of potential initial data values. In 3+1 relativity, initial data is given on a spacelike hypersurface at an instant in time. Although any arbitrary hypersurface could be chosen as initial data, only a hypersurface that satisfies the constraint equations is physically valid. Such a hypersurface is called a 'constraint hypersurface'. Imagine having a perfect numerical integration scheme with infinite machine accuracy. If we start with initial data exactly on the constraint hypersurface and integrate via the evolution equations we will stay on the constraint hypersurface for all time, giving us the physical results we sought. Now imagine the same perfect integrator, but this time we have initial data that is only correct to machine accuracy. Our perfect integrator will solve the evolution equations *exactly*. If the evolution equations are in such a form where the constraint hypersurface is a repeller, the solution will tend away from the hypersurface exponentially, quickly resulting in an unphysical solution. It is not the case that our integrator is giving us errors, rather it is following the trajectory exactly of the given initial data, which just happens to be exponentially diverging from the constraint hypersurface.

It is hard to see from inspection whether or not the evolution equations of the metric and extrinsic curvature have divergent behavior resulting in constraint violating modes. It is a little easier to investigate their inherent properties by instead studying the evolution equations of the constraints:

$$\partial_\perp H = 2\alpha K H - 4M^a D_a\alpha - 2\alpha D_a M^a \tag{6.3}$$

$$\partial_\perp M_a = -\frac{1}{2}\alpha D_a H - H D_a\alpha + \alpha K M_a \tag{6.4}$$

It is still not immediately obvious from the above equations what kind of behavior to expect from the evolution equations; however, if we do a couple of simplifications, we can show some examples of exponentially growing solutions. These are the constraint violating modes. We have written a spectral methods code[5] to evolve the constraint

[5]See chapter 7 for a summary of the code.

equations in (6.3) and (6.4). Consider the constraint evolution equations with $\alpha = 1$, $\beta = 0$, in one dimension with a flat metric:

$$\partial_t H = -2\partial_x M + 2KH \tag{6.5}$$
$$\partial_t M = -\frac{1}{2}\partial_x H + KM \tag{6.6}$$

The principal parts[6] of the above equations are:

$$\partial_t H = 2KH \tag{6.7}$$
$$\partial_t M = KM \tag{6.8}$$

with analytic solutions

$$lnH = ln(H_0) + 2Kt \tag{6.9}$$
$$lnM = ln(M_0) + Kt. \tag{6.10}$$

We evolve the coupled constraint evolution equations in the simplified form of equations (6.5) and (6.6) using K as a constant, periodic boundary conditions, and initial constraint violations of $M_0(x) = 0$ and $H_0(x)$ set to random noise of various amplitudes. Figure 6.1 shows that the evolution of the Hamiltonian constraint gives the expected form, as in equations (6.9). Figure 6.1 plots the L2norm of Hamiltonian constraint for initial H set to various amplitudes of random noise. It illustrates that a constraint violating mode will always grow at the expected rate, and achieving more accuracy in the initial data only delays the time it takes to reach a certain level of error in the constraints. Figure 6.2 shows the Hamiltonian constraint evolutions with random perturbations for initial conditions along with the analytic solution for the principal part (equation [6.9]). Next, figure 6.3 is a plot of the constraints as calculated from the constraint evolution equations and as calculated from the full evolution code. The evolution starts out as expected for a very short time and then grows super-exponentially. This illustrates that the exponentially growing modes can be a best case scenario. The constraints are always initially violated, even if just to

[6]For a second order linear partial differential equation $L(u) = Au_{tt} + Bu_{tx} + Cu_{xx} + Du_t + Eu_x + Fu = G$ where $u = u(x,t)$, $A = A(x,t)$, etc., the 'principal part' is $Au_{tt} + Bu_{tx} + Cu_{xx}$, unless $A = B = C = 0$, in which case the principal part if $Du_t + Eu_x$

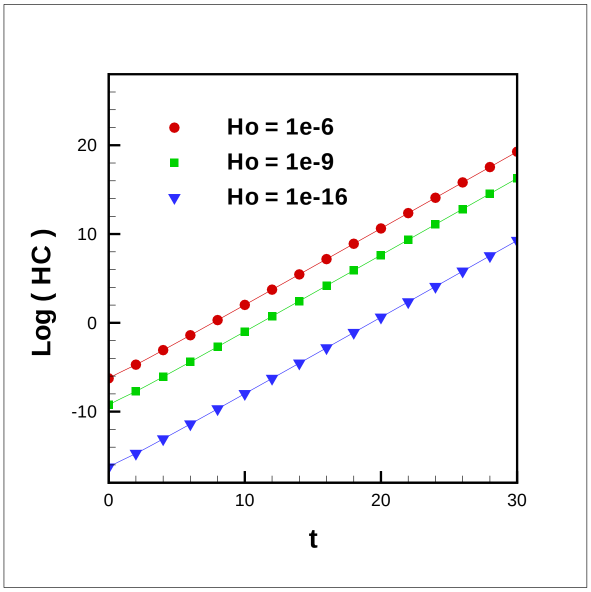

Figure 6.1: Exponential constraint growth from random initial perturbations of the Hamiltonian constraint. Plot shows the common logarithm of the L2 norm of the Hamiltonian constraint versus time for a 1D flat metric with $K = 1$ for initial values of Hamiltonian constraint $H_0 = 1e-6, 1e-9, 1e-16$.

machine accuracy, and that will cause exponential growth of constraint errors in the presence of constraint violating modes. During any evolution, more errors are introduced from the numerical techniques. Numerical integrators may also be unstable. These errors will necessarily be larger than the ones created from violating the initial conditions. This is one of the reasons that it took many years to focus on the problem of inherent instability of the Einstein equations; numerical experiments showed simulations lasting longer and behaving better with increased resolution and improved numerical integrators. However, better resolution and stable algorithms only succeed in following the exponentially growing solutions more accurately.

6.3 Constrained Evolution

If the initial value or constraint equations are satisfied initially, in theory they remain satisfied, but numerically they can never be exactly satisfied. If there are constraint violating modes, they are guaranteed to grow. One way of fixing this is

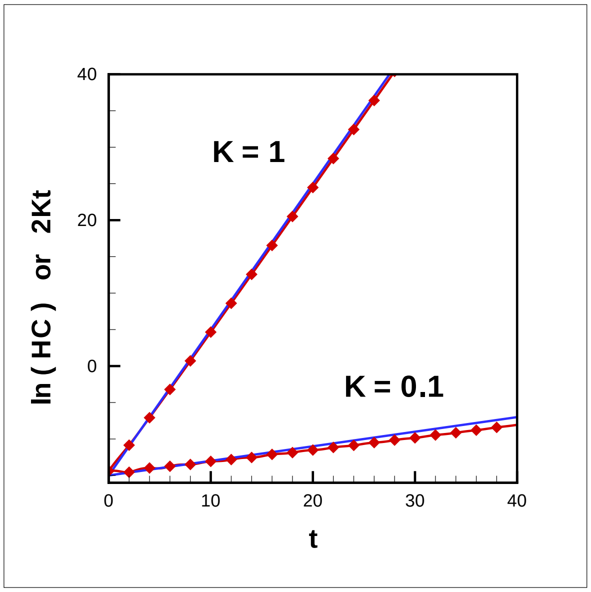

Figure 6.2: Exponential constraint growth from small initial perturbations of the Hamiltonian constraint. Plot shows the natural logarithm of the L2 norm of the Hamiltonian constraint versus time for a 1D flat metric with $K = 1$ or $K = 0.1$ for initial value of Hamiltonian constraint $H_0 = 1e - 9$. Also shown are the analytic solutions, $\ln(H) \sim 2Kt$. The numerical solution is the red dotted line, and the analytic solution is the smooth blue line.

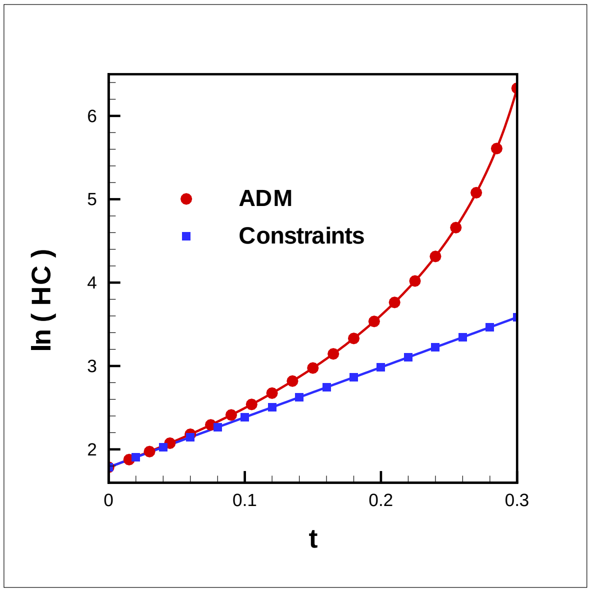

Figure 6.3: The blue (bottom) line shows the expected constraint error as calculated from the constraint evolution code. The red (top) line shows the constraint error when using the full evolution code. Both lines plot the natural logarithm of the L2 norm of the Hamiltonian constraint.

to modify the constraint evolution equations by adding multiples of the constraints to the evolution equations so that the constraint evolution equations have desired behavior. The constraint equations in vacuum are

$$H = K^2 - K_{ij}K^{ij} + R = 0 \tag{6.11}$$

$$M_i = D_j K_i^j - D_i K = 0. \tag{6.12}$$

You can freely add zeros to any equation, e.g.,

$$A + B = A + B + C$$

where $C = 0$. Since the constraints H,M_i are zero, they can be added in any multiples to the evolution equations in order to change their mathematical nature. Various groups have done this, with emphasis on attaining hyperbolicity in the evolution equations. Alternatively, one could focus on the form of the constraint evolution equations. As shown in section 6.2, the constraint evolution equations of the traditional ADM formalism have exponentially growing solutions. Chapter 8 shows a way to explicitly specify the form of the constraint evolution equations. Recall the simplified constraint evolution equations have principal part

$$\partial_t H = 2KH \tag{6.13}$$

$$\partial_t M = KM \tag{6.14}$$

with analytic solutions

$$lnH = ln(H_0) + 2Kt \tag{6.15}$$

$$lnM = ln(M_0) + Kt. \tag{6.16}$$

In our examples, we showed solutions with positive constant K. Negative constant K gives solutions that tend *toward* the constraint hypersurface. Equations (6.15) depend on the form of K, which generally changes during an evolution. This makes it hard to anticipate which kinds of data sets result in constraint violating modes. If we could specify the form of the constraint evolution equations, we could choose a form that would always make the constraint hypersurface an attractor. For example, we could insist that the constraint evolution equations take the form

$$\partial_t H = -c^2 H \tag{6.17}$$

$$\partial_t M = -d^2 M \tag{6.18}$$

where c and d are arbitrary constants. This gives exponentially decaying solutions

$$lnH = ln(H_0) - c^2 t \tag{6.19}$$

$$lnM = ln(M_0) - d^2 t. \tag{6.20}$$

That would ensure that solutions starting off of the constraint hypersurface would tend back towards the constraint hypersurface. In chapter 8, we describe in detail how to implement this idea for the purposes of performing constrained evolutions. An informal summary of the method is outlined below.

Consider a set of constraints $\mathcal{C}$ and a set of field variables φ:

$$\mathcal{C} \equiv (H, M^a) \tag{6.21}$$

$$\varphi \equiv (g_{\mu\nu}, K_{\mu\nu}) \tag{6.22}$$

The constraint evolution equations are given by

$$\dot{\mathcal{C}} = \frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi}). \tag{6.23}$$

Now let us split the basic variables φ into a subset of variables ϕ.

$$\dot{\mathcal{C}} = \frac{\delta\mathcal{C}}{\delta\phi}(\dot{\phi}) \tag{6.24}$$

To create new constraint evolution equations $\dot{\mathcal{C}}$, we form an equation relating the original constraint equations $\dot{\mathcal{C}}_{old}$ with $\dot{\mathcal{C}}_{new}$.

$$\dot{\mathcal{C}}_{old} = \frac{\delta\mathcal{C}}{\delta\phi}(\dot{\phi})_{old} \tag{6.25}$$

$$\dot{\mathcal{C}}_{new} = \frac{\delta\mathcal{C}}{\delta\phi}(\dot{\phi})_{new} \tag{6.26}$$

Subtracting the two gives

$$\Lambda = \frac{\partial\mathcal{C}}{\partial\phi}\Psi \tag{6.27}$$

where $\Lambda = \dot{\mathcal{C}}_{old} - \dot{\mathcal{C}}_{new}$ and $\Psi = \frac{\delta\mathcal{C}}{\delta\phi}(\dot{\phi})_{old} - \frac{\delta\mathcal{C}}{\delta\phi}(\dot{\phi})_{new}$. We know $\dot{\mathcal{C}}_{old}$, and we can choose $\dot{\mathcal{C}}_{new}$, giving us an expression for Λ. We then have an elliptic partial differential equation for the variables Ψ. Our original computer code solves the evolution equations $\dot{\varphi}_{old}$ with constraint evolution equations $\dot{\mathcal{C}}_{old}$. We want to modify these equations to have constraint evolution equations $\dot{\mathcal{C}}_{new}$. The new evolution equations $\dot{\varphi}_{new}$ can be calculated as follows:

$$\Lambda = \dot{\mathcal{C}}_{old} - \dot{\mathcal{C}}_{new} = \frac{\partial\mathcal{C}}{\partial\phi}\Psi$$

$$\dot{\mathcal{C}} = \frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi})$$

$$\Lambda = \frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi})_{old} - \frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi})_{new} = \frac{\partial\mathcal{C}}{\partial\phi}\Psi$$

$$\frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi})_{new} = \frac{\delta\mathcal{C}}{\delta\varphi}(\dot{\varphi})_{old} - \frac{\partial\mathcal{C}}{\partial\phi}\Psi$$

Our new evolution equations are

$$(\dot{\varphi})_{new} = (\dot{\varphi})_{old} - \frac{\delta\varphi}{\delta\phi}\Psi. \tag{6.28}$$

In the derivation of new evolution equations, a subset ϕ of the dynamic variables φ need to be specified. We choose the conformal variables so that

$$\phi = (\psi, w_a). \tag{6.29}$$

This will give us partial elliptic equations to solve for the variables

$$\Psi = (\Phi, W_a). \tag{6.30}$$

The term $\frac{\delta\varphi}{\delta\phi}\Psi$ is then

$$\frac{\delta\varphi}{\delta\phi}\Psi = \frac{\delta\varphi}{\delta\psi}\Phi + \frac{\delta\varphi}{\delta w_a}W_a. \tag{6.31}$$

Here we will give an example of a computation. Recall the conformal splitting of the metric $g_{ij} = \psi^4\tilde{g}_{ij}$. For $\varphi = g_{ij}$, this gives

$$\frac{\delta\varphi}{\delta\phi}\Psi = \frac{\delta g_{ij}}{\delta\psi}\Phi + \frac{\delta g_{ij}}{\delta w_a}W_a. \tag{6.32}$$

Calculating the derivatives

$$\frac{\delta g_{ij}}{\delta\psi}\Phi = 4\psi^3\tilde{g}_{ij}\Phi = \frac{4g_{ij}}{\psi}\Phi \tag{6.33}$$

and

$$\frac{\delta g_{ij}}{\delta w_a}W_a = 0 \tag{6.34}$$

and choosing $\psi = 1$, we use equations (6.28) and (6.32) to get a new evolution equation for g_{ij}

$$(\dot{g}_{ij})_{new} = (\dot{g}_{ij})_{old} - 4g_{ij}\Phi. \tag{6.35}$$

To calculate the Λ terms we have

$$\Lambda = \frac{\partial\mathcal{C}}{\partial\phi}\Psi \tag{6.36}$$

giving equations

$$\Lambda_0 = \frac{\delta H}{\delta\psi}\Phi + \frac{\delta H}{\delta W_a}W_a \tag{6.37}$$

$$\Lambda_a = \frac{\delta M_a}{\delta\psi}\Phi + \frac{\delta M_a}{\delta W_a}W_a. \tag{6.38}$$

Recall that we can get different forms of the conformal constraint equations using the different initial value decompositions as described in chapter 3. The elliptic partial differential equation we are left with after using the general form of the conformal constraints after a conformal transverse traceless decomposition (equation [3.13]) is

$$\Lambda_0 = -2\psi^{n+4}K^{ab}(LW)_{ab} - (2n+8)\frac{\Phi}{\psi}(K_{ab}K^{ab} - \frac{1}{3}K^2) - 4R\frac{\Phi}{\psi} - 8D^2(\frac{\Phi}{\psi}) \quad (6.39)$$

$$\begin{aligned}\Lambda_a = & -(n+10)\psi^{-2}D_b\psi(K_a^b - \frac{1}{3}\delta_a^b K\Phi) + \psi^{-1}(n+10)(K_a^b - \frac{1}{3}\delta_a^b K)D_b\Phi \quad (6.40)\\ & +\psi^{-1}(n+4)D_b(K_a^b - \frac{1}{3}\delta_a^b K)\Phi + (n+4)\psi^{n+3}D_b\psi(LW)_a^b + \psi^{n+4}D_b(LW)_a^b\end{aligned}$$

Choosing $\psi = 1$ we have:

$$\Lambda_0 = -2K^{ab}(LW)_{ab} - (2n+8)\Phi(K_{ab}K^{ab} - \frac{1}{3}K^2) - 4R\Phi - 8D^2(\Phi) \quad (6.41)$$

$$\begin{aligned}\Lambda_a = & (n+10)(K_a^b - \frac{1}{3}\delta_a^b K)D_b\Phi\\ & +(n+4)D_b(K_a^b - \frac{1}{3}\delta_a^b K)\Phi + D_b(LW)_a^b \quad (6.42)\end{aligned}$$

If we then use the usual conformal transverse traceless decomposition ($n = -10$), the Λ equations are

$$\Lambda_0 = -2K^{ab}(LW)_{ab} + 12\Phi(K_{ab}K^{ab} - \frac{1}{3}K^2) - 4R\Phi - 8D^2(\Phi) \quad (6.43)$$

$$\Lambda_a = -6D_b(K_a^b - \frac{1}{3}\delta_a^b K)\Phi + D_b(LW)_a^b. \quad (6.44)$$

If we instead use a 'same sign' conformal transverse traceless decomposition ($n = -4$), we get

$$\Lambda_0 = -2K^{ab}(LW)_{ab} - 4R\Phi - 8D^2(\Phi) \quad (6.45)$$

$$\Lambda_a = 6(K_a^b - \frac{1}{3}\delta_a^b K)D_b\Phi + D_b(LW)_a^b. \quad (6.46)$$

Chapter 8 uses the equations (6.43) and (6.44). The same sign conformal transverse traceless equations were calculated to test whether they would produce elliptic equations which were easier to solve. Preliminary numerical tests show that the original Λ equations converge faster than the same sign Λ equations. The details of the code are described in chapter 7.

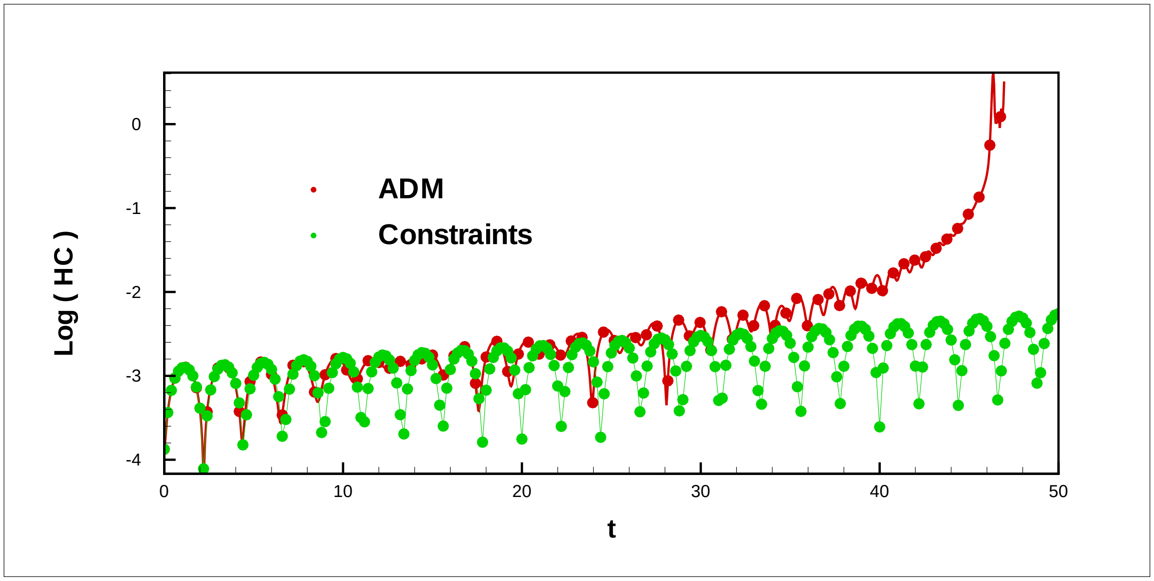

Figure 6.4: Full evolution code and constraint code until evolution code crash.

Chapter 8 shows a figure displaying the results of a constrained vs. unconstrained evolution. Figure 6.4 shows the results of the same test case of the unconstrained evolution along with the evolution of the constraints until the evolution code crashes.

Chapter 7

Spectral Methods

7.1 Introduction

The majority of numerical relativity codes use what is called 'finite differencing' to estimate or calculate functions and derivatives. Finite differencing is usually straightforward to implement and fast to compute per grid point. There are a few drawbacks to using finite differencing. Boundary conditions can be tricky, because the standard finite difference stencils (grid points used in calculations) usually extend beyond the domain of the problem. This entails defining 'ghost zones' and a possible need for interpolating schemes. Another drawback is the memory requirements for acceptable accuracy. Convergence is $\approx \frac{1}{N^k}$ where N is the number of grid points, so the accuracy increases to the power of the order of the differencing scheme. Higher order schemes can decrease memory requirements, but make the boundary issues more complex. Higher order will call either for extra ghost zones to provide the required stencil, or special (and usually complex) treatment of differencing at the boundaries.

Spectral methods[1] can be of great benefit for memory intensive problems. Spectral methods have exponential convergence, a convergence of $\approx \frac{1}{N^N}$, Exponential convergence means that for a smooth function, the error will decay faster than $\frac{1}{N^k}$ for any k. Convergence in finite differencing is only $\approx \frac{1}{N^k}$, so to achieve a given accuracy, spectral

[1]Throughout this dissertation, 'spectral methods' should be taken to mean 'pseudospectral collocation methods'. For a description of spectral methods in a more general sense, see for example [53].

methods require significantly less grid points. Spectral methods are not necessarily faster than finite differencing because of the time it takes to calculate the functions and derivatives. A finite difference calculation usually requires a few multiplies and additions per grid point, while a spectral methods calculation must be taken over the entire grid and can involve using trigonometric functions which require significantly more floating point operations than even a complex finite difference stencil. Spectral methods can sometimes have an advantage over finite differencing by of allowing one to give more precise boundary conditions. For a finite difference code, one generally has to estimate the boundaries (via finite differenced derivatives, and perhaps using ghost zones), while in spectral methods, one can specify the boundaries exactly for the boundary grid points.

In principle, any evolution code that is written in a somewhat object oriented fashion can substitute finite difference calculations with spectral method calculations. One would ideally have a subroutine that calculates derivatives based on a given function and grid structure, and then substitute the finite differencing routine with a spectral methods routine. The actual integration can be performed exactly the same via the 'method of lines' (finite differencing in time). In practice, however, spectral methods bring their own host of complications to a calculation. It is generally more difficult to program and in particular, to parallelize in a straightforward manner[2]. One of the problems with using spectral methods for general relativity is that spectral methods are suited for smooth functions. Relativity problems often have steep slopes or singularities. To use spectral methods for non smooth problems, special domain decompositions are needed, which adds complications with parallelization, stability and boundary implementation. There is also a problem known as 'spectral blocking' which is the tendency for garbage to build up in the high frequency modes. Spectral blocking can be addressed with a technique known as 'filtering', but there is no set prescription for implementing it.

[2]Pseudospectral collocation methods may be harder to parallelize as they have basis functions defined over the entire grid. Spectral elements have basis functions defined over subsections of the grid, and they are therefore parallelized as well or better than with finite differencing.

7.2 Comparison of Finite Differencing and Spectral Methods– Example in 1D

The most common way of solving a differential equation is with finite differencing. In this section, an example problem relating the differences between spectral methods and finite differencing will be presented. The aim is to transition someone already familiar with finite differencing into the vernacular of spectral methods. It may be helpful to someone who is new to finite differencing as well.

Consider the problem of solving the one-dimensional heat equation for $u \equiv u(x,t)$ described by

$$u_t = u_{xx}, \qquad (x_L < x < x_R,\ 0 < t < \infty) \tag{7.1}$$

where $u_t = \partial u(x,t)/\partial t$ and $u_{xx} = \partial^2 u(x,t)/\partial x^2$ with Dirichlet boundary conditions at left and right boundaries x_L and x_R as

$$u(x_L,t) = g(t) \qquad u(x_R,t) = h(t) \tag{7.2}$$

and initial condition

$$u(x,0) = f(x). \tag{7.3}$$

7.2.1 Heat Equation Using Finite Differencing

(I) First we define the function u at the initial time, which is given by initial condition $u(x,0) = f(x)$. We assume that we are given or can generate the values of $f(x)$ at $t = 0$ at points along the grid $x = \{x_i\}$.

(II) Next we need to define the spatial grid $x = \{x_i\}$, where there are $N+1$ grid points, the grid size is $(x_R - x_L)$ and the grid resolution is $\Delta x = (x_R - x_L)/N$. Our grid is then

$$\{x_i\} = x_L + i\Delta x \qquad (i = 0, 1, ..., N-1, N) \tag{7.4}$$

(III) Next we define a spatial differentiation operator on u, for example:

$$u_{xx}(x_i,t) \equiv \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{(\Delta x)^2} + O((\Delta x)^2) \tag{7.5}$$

(IV) Next we define a time differentiation operator and choose a numerical integrator. The simplest example is forward Euler, so that

$$u_t \equiv \frac{u(t+1) - u(t)}{\Delta t} \tag{7.6}$$

where t is the current timestep. The timestep Δt chosen will depend on the numerical integrator chosen and will usually also depend on the spatial resolution.

(V) Take a timestep. Solve for the new solution u_{new} at $t+1$ by using the values of the solution u_{old} at the current [3]timestep t. For forward Euler,

$$u_{new} = u_{old} + RHS(u_{old})\Delta t \tag{7.7}$$

where in this case

$$RHS(u_{old}) = \frac{u_{old}(x_{i+1}) - 2u_{old}(x_i) + u_{old}(x_{i-1})}{(\Delta x)^2} \tag{7.8}$$

(VI) Enforce boundary conditions at $u_{new}(x_L)$ and $u_{new}(x_R)$.

7.2.2 Heat Equation Using Spectral Methods

(I) First we define the function u at the initial time, which is given by initial condition $u(x, 0)$. Assume that the function $u(x)$ can be represented exactly by using a set of basis functions $\psi_n(x)$.

$$u(x) = \sum_{n=0}^{\infty} a_n \psi_n(x), \tag{7.9}$$

and that the function $u(x)$ can be well approximated by truncating the series at sufficiently high N as

$$\tilde{u}(x) \approx \sum_{n=0}^{N} a_n \psi_n(x). \tag{7.10}$$

[3]This is how an explicit integrator is performed. If an implicit scheme is used, u_{new} could also depend on values at u_{new}.

(II) Define a grid

$$x = \{x_n\} \quad (n = 0, 1..., N-1, N) \tag{7.11}$$

such that the $\{x_n\}$ are extrema (or alternatively the zeros) of the basis functions. We know the basis functions ψ_n and can calculate the coefficients a_n at the collocation points $\{x_n\}$ for the initial function $u(x, 0)$.

(III) Define the spatial differentiation operator on u. The derivative operator $\partial^2/\partial x^2$ is calculated exactly, that is, the exact derivative of the approximate function $\tilde{u}(x)$ is taken.

(IV) Define a time differentiation operator and choose a numerical integrator. This is done in the same way as in section 7.2.1. We know the function $u(x_n)$ at each grid point x_n, and we integrate each grid point using finite differencing in time rather than using a spectral decomposition in time. This technique is known as 'the method of lines'.

(V) Take a timestep (as in section 7.2.1).

(VI) Enforce boundary conditions, given exactly at $u(x_0)$ and $u(x_N)$.

7.3 Details from our code

In this section, we will describe spectral methods as applied to our ADM evolution code. There are two versions of the code, one using a Chebyshev basis and one using a Fourier basis. Here we will describe the Chebyshev code. Instead of including references throughout, let us list them here. We have found the following references invaluable in implementing our code: Numerical Recipes by Press et. al. [54], Chebyshev and Fourier Spectral Methods by J.P. Boyd [53], and the existing relativity spectral methods papers including [55, 56].

The evolution equations are partial differential equations with first order time derivatives of the unknown function $u \equiv u(x, t)$. The first time derivative of t will be written as $\partial_t u \equiv \partial u(x, t)/\partial t$, and it is a function of u and its spatial derivatives such that

$$\partial_t u = F(u, \partial_x u, ...). \tag{7.12}$$

We are solving for our unknown function u. To solve for the time dependence part of our equation (7.12), we use the method of lines. The 'method of lines' is a way of treating a partial differential equation like an ordinary differential equation. If the set of equations can be written as an initial value problem, then the method of lines can be used. In our case, we define the problem our grid and then we calculate the function F in equation (7.12) at each grid point using spectral methods. We then integrate the u at each gridpoint as usual, with finite differencing in time. The numerical integrator we use in the code is a fourth order Runge–Kutta method.

To evaluate the right hand side of equation (7.12) at a constant timestep (so that u is only a function of x), we express $u \equiv u(x)$ and its spatial derivatives using spectral methods. First we choose a basis set and a grid. We choose the Chebyshev polynomials T_k as our basis set. We choose the grid x_k to be a set of collocation points of our basis function. There are two choices for collocation points. Gauss–Chebyshev collocation points are the zeros of the Chebyshev polynomials T_n and Gauss–Lobatto collocation points are the extrema and endpoints of T_n. Using Gauss–Lobatto allows us to have collocation points directly on the boundaries, making the specification of boundary conditions more straightforward. For Gauss–Lobatto collocation, the grid points are

$$x_k = cos\left(\frac{\pi k}{N}\right) \qquad k = 0, 1, ..., N \tag{7.13}$$

We use spectral methods mainly to evaluate spatial derivatives. Assuming we know the initial function $u(x, t = 0)$ at the collocation points $\{x_k\}$, we can find the coefficients c_k of its Chebyshev expansion

$$u(x) \approx \sum_{k=0}^{N} c_k T_k(x). \tag{7.14}$$

The coefficients can be calculated in terms of cosines as

$$c_k = \alpha_k \sum_{j=0}^{N} \beta_j u\left(cos\left(\frac{\pi j}{N}\right)\right) cos\left(\frac{\pi j k}{N}\right) \tag{7.15}$$

where

$$
\begin{aligned}
\alpha_k &= \frac{1}{N} \quad \text{for } \; k = 0, \; k = N && (7.16)\\
\alpha_k &= \frac{2}{N} \quad \text{for } \; k \neq 0, \; k \neq N \\
\beta_j &= \frac{1}{2} \quad \text{for } \; j = 0, \; j = N \\
\beta_j &= 1 \quad \text{for } \; j \neq 0, \; j \neq N
\end{aligned}
$$

(7.17)

We use routines modified from Numerical Recipes to do this. (Modified because Numerical Recipes routines are written with Gauss–Chebyshev collocation points in mind, while we use Gauss–Lobatto.) The NR routine for finding Chebyshev coefficients based on given values of a function is called `chebft` (for Chebyshev fit). Once the coefficients are known, recursion relations can be used to find derivatives based on the coefficients. For this we use a modified `chder` subroutine. The derivative relationship is

$$
\begin{aligned}
c'_N &= 0 && (7.18)\\
c'_{N-1} &= 2Nc_N \\
c'_k &= c'_{k+2} + 2(k+1)c_{k+1} \\
c'_0 &= \frac{1}{2}(c'_2 + 2c_1)
\end{aligned}
$$

The derivative routine takes the coefficients c_k of a Chebyshev approximated function $u(x)$ and returns the coefficients c'_k of the derivative of the Chebyshev approximated function $u_x(x)$. To get the function values of $u_x(x)$ we use

$$u_x(x) \approx \sum_{k=0}^{N} c'_k T_k(x) \tag{7.19}$$

which is performed via a modified version of the Numerical Recipes `chebeval`, or Chebyshev evaluation.

Chebyshev polynomials are defined on an interval $[-1, 1]$, and we can use the linear mapping

$$y = \frac{x - \frac{1}{2}(b+a)}{\frac{1}{2}(b-a)} \tag{7.20}$$

to take a range $[a, b]$ and map it into the required range $[-1, 1]$.

For explicit integrators, we need a Courant factor proportional to the spatial grid resolution. Our collocation grid has variable spacing, and the smallest grid spacing is used when calculating the Courant condition. The smallest grid spacing for the Gauss–Lobatto grid is $\Delta x = 1 - cos(x)$ which is approximately $\frac{1}{2}(\pi^2/N^2)$.

7.4 Summary

Using spectral methods has allowed us to have an evolution code small enough to fit on a desktop computer using one processor. While this was useful for some testing, it was determined that there was not enough time or personnel to address fixing the problems that came up. To utilize the spectral code for more realistic problems, the issues of appropriate boundary conditions, domain decomposition, spectral blocking, and parallelization would have to be addressed.

In theory, spectral methods are ideally suited for proper boundary conditions, as a boundary condition can be specified 'exactly'. In practice, there are other issues that complicate the matter. The simplest spectral methods test case solver will use a Fourier basis to avoid the boundary condition issue altogether. A Fourier basis implements periodic boundary conditions. The problem with this is that periodic boundary conditions are unrealistic for most scenarios. A Chebyshev basis is considered to be the best choice in any general case [53], but in terms of the location of the collocation points, Chebyshev polynomials give the opposite of what would be ideal for boundary conditions. A Chebyshev basis has collocation points bunched up at the ends. In relativity scenarios, one generally wants low resolution at boundaries (where space is mostly flat) and higher resolution in the middle. A basis of Rational Chebyshev Functions is better suited to asymptotically flat spacetime boundaries, so this basis could be used. While a simple change of basis in the code may be straightforward, there is a lack of experience with using these Rational Chebyshev functions and unexpected complications will undoubtedly arise.

Spectral methods have excellent convergence properties IF the functions they are representing are analytic, or at the very least 'smooth'[4]. The problem of interest in relativity is the black hole problem—obviously far from smooth. Evolving a 'puncture' is out of the question with single domain spectral methods. Multiple domains would need to be used. In a multiple domain spectral code, one would break up the problem into domains, and perhaps have inner boundary conditions such as the function being solved outside the apparent horizon. While this is possible, and in fact has been implemented by various groups [55, 56], it is not a realistic prospect for our small group to undertake in the near future.

A major problem in spectral methods is the phenomenon known as 'spectral blocking', which is the tendency for garbage to build up within high N coefficients[5] One of the solutions to spectral blocking is to use filtering, which sets the coefficients of the highest modes to zero. To our knowledge, and in our experience, there are some guidelines but no foolproof established formula for successful filtering. A study and some numerical experimentation in filtering for each specific situation would be necessary in building an efficient spectral method solver, unfortunately yet again another time consuming project.

Building up a numerical relativity spectral methods code from scratch might be an interesting prospect, but not realistic in the foreseeable future. One possibility for a future project is to try to use LORENE [59, 60]. LORENE is a grid infrastructure (like PARAMESH or HAD) for spectral methods. At the time of this writing, the grid structure in LORENE is confined to a spherical grid (spherical coordinates using a sphere and optional concentric shells). It might be prudent to wait for the developers of LORENE to implement a Cartesian grid option before attempting to use it for our purposes.

[4]Loosely speaking, a function is analytic if it can be represented as a Taylor series, and a function is smooth if all partials of all orders exist and are continuous [57, 58]

[5]A detailed analysis of aliasing, blocking and filtering is given in Boyd [53].

Chapter 8

Modifying the Einstein equations off the constraint hypersurface

J. David Brown and Lisa L. Lowe
Modifying the Einstein equations off the constraint hypersurface
Physical Review D

PHYSICAL REVIEW D **74,** 104023 (2006)

Modifying the Einstein equations off the constraint hypersurface

J. David Brown and Lisa L. Lowe

Department of Physics, North Carolina State University, Raleigh, North Carolina 27695, USA

(Received 6 June 2006; published 20 November 2006)

A new technique is presented for modifying the Einstein evolution equations off the constraint hypersurface. With this approach the evolution equations for the constraints can be specified freely. The equations of motion for the gravitational field variables are modified by the addition of terms that are linear and nonlocal in the constraints. These terms are obtained from solutions of the linearized Einstein constraints.

DOI: 10.1103/PhysRevD.74.104023 PACS numbers: 04.25.Dm

The Einstein equations separate into a set of evolution equations and a set of constraints. The evolution equations are partial differential equations (PDE's) that determine how the gravitational field variables g_{ab} (the spatial metric) and K_{ab} (the extrinsic curvature) evolve forward in time. The constraint equations are PDE's that the field variables must satisfy at each instant of time. From a Hamiltonian point of view, the evolution equations define solution trajectories in phase space with coordinates g_{ab} and momenta K_{ab}. Physical trajectories are those that lie in the constraint hypersurface, or subspace, of the gravitational phase space.

Einstein's theory of gravity is a "first class" theory, that is, the time derivatives of the constraints are linear combinations of the constraints. This property implies that, analytically, the constraints will hold at each instant of time if they hold at the initial time. However, for numerically generated solutions of the theory the initial data will not satisfy the constraints precisely and numerical errors will kick the phase space trajectory away from the constraint hypersurface. This is a critical problem for numerical modeling because the Einstein evolution equations, as they are usually written, admit solutions that rapidly diverge away from the constraint hypersurface [1,2]. Any numerical scheme that evolves the gravitational field data using the evolution equations in one of their traditional forms will eventually fail to produce physically meaningful results. Inevitably the numerical solution will choose to follow a trajectory that violates the constraints.

A number of strategies have been devised to address this problem. One approach is to modify the theory off the constraint hypersurface by adding linear combinations of constraints to the evolution equations [1,3–7]. In this way, one hopes to alter the solution trajectories so that they are better behaved away from the constraint hypersurface. We will use the terminology "off-shell" to refer to solution trajectories that lie off the constraint hypersurface.

The strategy discussed in this paper is of this sort. We add terms proportional to the constraints to the Einstein evolution equations in such a way that the evolution equations for the constraints can be freely specified. In principle, we can eliminate all constraint violating modes by demanding, for example, that the time derivatives of the constraints should vanish. The price we pay for this degree of control over the unphysical, off-shell solutions is that the terms added to the evolution equations are nonlocal. They are determined through the solution of an elliptic system of PDE's.

Another strategy for keeping a numerically generated solution from diverging away from the constraint hypersurface is constrained evolution. In this scheme the constraints are used in place of certain evolution equations to update some of the gravitational field variables in time. This approach has worked well for spherically and axisymmetric problems [8–12]. A closely related idea is constraint projection [5,13,14]. With constraint projection, one evolves the full set of field variables using the evolution equations, then periodically (perhaps every time step) solves the constraints to project the solution back to the constraint hypersurface. Both constrained evolution and constraint projection require the solution of the constraint equations during the course of evolution. For these approaches to be viable, the constraints must be expressed as an elliptic system of PDE's. From a computational perspective, our strategy is closely related to constraint projection since we also solve an elliptic system of PDE's at every (or nearly every) time step. In fact, the PDE's that we solve are the linearized Einstein constraints.

It will be useful to give an overview of our procedure for modifying the off-shell solutions in a formal, general context. Consider a theory, like general relativity, described by a set of first class constraints $\mathcal{C}_A$. Let ψ_μ denote the basic field variables. These variables satisfy first order in time differential equations of motion, $\dot\psi_\mu = (\dot\psi_\mu)_{\text{old rhs}}$, where the "old right-hand sides" $(\dot\psi_\mu)_{\text{old rhs}}$ are functions of ψ_μ and their spatial derivatives. We have included the descriptor "old" since we will soon create "new" right-hand sides by adding functions of the constraints. The evolution equations for the constraints are obtained from the evolution equations for the ψ's by differentiating the constraints in time. This yields $\dot{\mathcal{C}}_A = (\dot{\mathcal{C}}_A)_{\text{old rhs}}$ where the right-hand sides (rhs's) are given by

$$(\dot{\mathcal{C}}_A)_{\text{old rhs}} = \frac{\delta\mathcal{C}_A}{\delta\psi_\mu}(\dot\psi_\mu)_{\text{old rhs}}. \tag{1}$$

The expression $\delta\mathcal{C}_A/\delta\psi_\mu$ is the Fréchet derivative of the constraints with respect to the field variables [15]. It satisfies $\mathcal{C}_A(\psi+\sigma)-\mathcal{C}_A(\psi)=(\delta\mathcal{C}_A/\delta\psi_\mu)\sigma_\mu$, in the limit as the norm of σ_μ goes to zero. (ψ_μ and σ_μ are defined as vectors in a suitable Banach space.) If $\mathcal{C}_A$ depends on spatial derivatives of ψ_μ, then the Fréchet derivative is a differential operator.

Now let us split the basic variables ψ_μ into two sets, ϕ_A and χ_i. Note that there are as many ϕ's as there are constraints. With this splitting, Eq. (1) becomes

$$(\dot{\mathcal{C}}_A)_{\text{old rhs}}=\frac{\delta\mathcal{C}_A}{\delta\phi_B}(\dot\phi_B)_{\text{old rhs}}+\frac{\delta\mathcal{C}_A}{\delta\chi_i}(\dot\chi_i)_{\text{old rhs}}. \quad (2)$$

Next, we replace the old equations of motion for the ϕ's with new equations of motion. This leads to new equations of motion for the constraints,

$$(\dot{\mathcal{C}}_A)_{\text{new rhs}}=\frac{\delta\mathcal{C}_A}{\delta\phi_B}(\dot\phi_B)_{\text{new rhs}}+\frac{\delta\mathcal{C}_A}{\delta\chi_i}(\dot\chi_i)_{\text{old rhs}}. \quad (3)$$

By subtracting the previous two results, we find

$$\Lambda_A=\frac{\partial\mathcal{C}_A}{\partial\phi_B}\Phi_B, \quad (4)$$

where Φ_A is the difference between the new and old rhs's for the variables ϕ_A, $\Phi_A\equiv(\dot\phi_A)_{\text{new rhs}}-(\dot\phi_A)_{\text{old rhs}}$, and Λ_A is the difference between the new and old rhs's for the constraints, $\Lambda_A\equiv(\dot{\mathcal{C}}_A)_{\text{new rhs}}-(\dot{\mathcal{C}}_A)_{\text{old rhs}}$. In terms of the original field variables ψ_μ, the new equations of motion are

$$(\dot\psi_\mu)_{\text{new rhs}}=(\dot\psi_\mu)_{\text{old rhs}}+\frac{\delta\psi_\mu}{\delta\phi_A}\Phi_A. \quad (5)$$

Now we turn the reasoning around. We do not actually choose new equations of motion for the ϕ's. Instead, we specify new evolution equations for the constraints by freely choosing the expressions $(\dot{\mathcal{C}}_A)_{\text{new rhs}}$. The functions Λ_A are then determined, and Eqs. (4) are solved for Φ_A. The new equations of motion for the original field variables are given by Eqs. (5).

Because the theory is first class, $(\dot{\mathcal{C}}_A)_{\text{old rhs}}$ is a linear combination of constraints. Let us choose $(\dot{\mathcal{C}}_A)_{\text{new rhs}}$ to be a linear combination of constraints as well. Then Λ_A is a linear combination of constraints and, according to Eq. (4), Φ_A is a (possibly nonlocal) linear combination of constraints. It follows that the new equations of motion for ψ_μ differ from the old equations by a linear combination of constraints.

Equations (4) are the linearized constraints. To be precise, consider a field configuration ψ_μ that does not satisfy the constraints and let $\bar\psi_\mu=\psi_\mu-\Psi_\mu$. If $\bar\psi_\mu$ satisfies the constraints then to linear order, Ψ_μ satisfies $\mathcal{C}_A=(\delta\mathcal{C}_A/\delta\psi_\mu)\Psi_\mu$. This is Eq. (4) with $\Psi_\mu=(\delta\psi_\mu/\delta\phi_A)\Phi_A$ and $\mathcal{C}_A=\Lambda_A$.

With the procedure outlined above, we can freely specify the rhs's of the constraint evolution equations, as long as they are a linear combination of constraints. We then solve Eq. (4) for Φ_A and modify the equations of motion for the ψ's to Eq. (5). In this way we leave the equations of motion for the basic field variables unchanged on the constraint hypersurface, but we can modify their off-shell form to eliminate the constraint violating modes.

Let us apply this formalism to general relativity. The basic field variables (the ψ's) are the spatial metric g_{ab} and the extrinsic curvature K_{ab}. The equations of motion as written by York [16] are $\partial_\perp g_{ab}=(\partial_\perp g_{ab})_{\text{old rhs}}$ and $\partial_\perp K_{ab}=(\partial_\perp K_{ab})_{\text{old rhs}}$, where

$$(\partial_\perp g_{ab})_{\text{old rhs}}=-2\alpha K_{ab}, \quad (6a)$$
$$(\partial_\perp K_{ab})_{\text{old rhs}}=\alpha(KK_{ab}-2K_{ac}K^c_b+R_{ab})-D_aD_b\alpha. \quad (6b)$$

Here, α is the lapse function, D_a is the spatial covariant derivative, and R_{ab} is the spatial Ricci tensor. The operator $\partial_\perp\equiv\partial_t-\mathcal{L}_\beta$ is the difference between the time derivative and the Lie derivative along the shift vector β^a. This operator plays the role of the "dot" in the formal analysis.

The constraints for general relativity are

$$\mathcal{H}\equiv K^2-K_{ab}K^{ab}+R, \quad (7a)$$
$$\mathcal{M}_a\equiv D_bK^b_a-D_aK. \quad (7b)$$

With the equations of motion (6), we find

$$(\partial_\perp\mathcal{H})_{\text{old rhs}}=2\alpha K\mathcal{H}-2\alpha D_a\mathcal{M}^a-4\mathcal{M}^aD_a\alpha, \quad (8a)$$
$$(\partial_\perp\mathcal{M}_a)_{\text{old rhs}}=\alpha K\mathcal{M}_a-\mathcal{H}D_a\alpha-\alpha D_a\mathcal{H}/2 \quad (8b)$$

for the rhs's of the constraint evolution equations.

In order to proceed, we must select a subset of the variables g_{ab}, K_{ab} to play the role of the ϕ's. We will use the conformal transverse-traceless decomposition developed by Lichnerowicz and York for solving the initial value problem (see, for example, Ref. [17]). To begin, we split the metric and extrinsic curvature into

$$g_{ab}=\varphi^4\tilde g_{ab}, \quad (9a)$$
$$K_{ab}=\varphi^{-2}A_{ab}+\tfrac{1}{3}\varphi^4\tilde g_{ab}\tau, \quad (9b)$$

where φ is the conformal factor, $\tilde g_{ab}$ is the conformal metric, τ is the trace of the extrinsic curvature, and A_{ab} is symmetric and trace free. Note that these definitions are invariant under the conformal transformation [18,19] $\tilde g_{ab}\to\xi^4\tilde g_{ab}$, $\varphi\to\xi^{-1}\varphi$, $A_{ab}\to\xi^{-2}A_{ab}$, $\tau\to\tau$.

The tensor A_{ab} can be decomposed in terms of a symmetric, transverse, traceless tensor B_{ab} and a vector w_a,

$$A_{ab}=(\tilde{\mathbb{L}}w)_{ab}+B_{ab}. \quad (10)$$

The operator $\tilde{\mathbb{L}}$ is defined by $(\tilde{\mathbb{L}}w)_{ab} \equiv \tilde{D}_a w_b + \tilde{D}_b w_a - 2\tilde{g}_{ab}\tilde{D}_c w^c/3$. It follows that w_a satisfies the elliptic equation $\tilde{D}^b(\tilde{\mathbb{L}}w)_{ab} = \tilde{D}^b A_{ab}$. The conformal transformation rule for B_{ab} is $B_{ab} \to \xi^{-2}B_{ab}$ and the rule for w_a is defined by the relation $(\tilde{\mathbb{L}}w)_{ab} \to \xi^{-2}(\tilde{\mathbb{L}}w)_{ab}$.

Let φ and w_a play the role of the ϕ's in Eqs. (4) and (5). The Fréchet derivatives that appear in those equations are somewhat tedious but straightforward to compute. If we let Φ and W_a denote the unknowns (the Φ_A's) in Eq. (4), we find the following results:

$$\Lambda_0 = -8D^2(\Phi/\varphi) - 2K^{ab}(\tilde{\mathbb{L}}W)_{ab}/\varphi^2 - [4K^2 - 12K_{ab}K^{ab} + 4R]\Phi/\varphi, \tag{11a}$$

$$\Lambda_a = D^b[(\tilde{\mathbb{L}}W)_{ab}/\varphi^2] - 6(D^b K_{ab} - D_a K/3)\Phi/\varphi. \tag{11b}$$

Here, Λ_0 and Λ_a are the differences between the new and old rhs's of the evolution equations for the constraints:

$$\Lambda_0 \equiv (\partial_\perp \mathcal{H})_{\text{new rhs}} - (\partial_\perp \mathcal{H})_{\text{old rhs}}, \tag{12a}$$

$$\Lambda_a \equiv (\partial_\perp \mathcal{M}_a)_{\text{new rhs}} - (\partial_\perp \mathcal{M}_a)_{\text{old rhs}}. \tag{12b}$$

The unknowns in Eqs. (11) are the differences between the new and old rhs's for the φ and w_a equations of motion. From the formal Eq. (5), we find the new equations of motion:

$$(\partial_\perp g_{ab})_{\text{new rhs}} = (\partial_\perp g_{ab})_{\text{old rhs}} + 4g_{ab}\Phi/\varphi, \tag{13a}$$

$$(\partial_\perp K_{ab})_{\text{new rhs}} = (\partial_\perp K_{ab})_{\text{old rhs}} + (\tilde{\mathbb{L}}W)_{ab}/\varphi^2 - 2(K_{ab} - Kg_{ab})\Phi/\varphi \tag{13b}$$

for the metric and extrinsic curvature.

Let us assume that the new equations of motion for φ and w_a, like the old, are conformally invariant. Then we see that Φ and W_a inherit the conformal transformation properties $\Phi \to \xi^{-1}\Phi$ and $(\tilde{\mathbb{L}}W)_{ab} \to \xi^{-2}(\tilde{\mathbb{L}}W)_{ab}$. It follows that Eqs. (11) and (13) are invariant under conformal transformations. In other words, these equations hold for any choice of splitting of the physical metric g_{ab} into a conformal factor φ^4 and conformal metric $\tilde{g}_{ab}$. For simplicity, we can choose the conformal factor to be unity, $\varphi = 1$, and the conformal metric to coincide with the physical metric, $\tilde{g}_{ab} = g_{ab}$. Then Eqs. (11) become

$$\Lambda_0 = -8D^2\Phi - 2K^{ab}(\mathbb{L}W)_{ab} - [4K^2 - 12K_{ab}K^{ab} + 4R]\Phi, \tag{14a}$$

$$\Lambda_a = D^b(\mathbb{L}W)_{ab} - 6(D^b K_{ab} - D_a K/3)\Phi. \tag{14b}$$

This is an elliptic system of linear PDE's for Φ and W_a. The new equations of motion (13) become

$$(\partial_\perp g_{ab})_{\text{new rhs}} = (\partial_\perp g_{ab})_{\text{old rhs}} + 4g_{ab}\Phi, \tag{15a}$$

$$(\partial_\perp K_{ab})_{\text{new rhs}} = (\partial_\perp K_{ab})_{\text{old rhs}} + (\mathbb{L}W)_{ab} - 2(K_{ab} - Kg_{ab})\Phi. \tag{15b}$$

This is our main result. We can now specify new rhs's for the Hamiltonian and momentum constraint equations of motion. The Λ's are found from Eqs. (12), and Eqs. (14) are solved for Φ and W_a. These results are used in the new equations of motion, Eqs. (15). Note that these equations contain only the physical metric and extrinsic curvature—all references to the conformal transverse-traceless splitting have disappeared.

What follows is a numerical demonstration that Eqs. (14) and (15) allow us the freedom to prescribe the evolution of the constraints by altering the Einstein equations off-shell. For this test we use periodic identification in the x, y, and z coordinate directions with periods of 2π. Thus, for now, we intentionally avoid facing the very important issue of boundary conditions. We use initial data that violates the constraints. The spatial metric is flat with Cartesian coordinates, $g_{ab} = \delta_{ab}$. The diagonal components of the extrinsic curvature are given by $K_{xx} = K_{yy} = K_{zz} = A/3$. The off-diagonal component K_{xy} is

$$K_{xy} = \varepsilon_1 \cos^2(z) + \varepsilon_2 \cos(x)\cos(y). \tag{16}$$

The components K_{yz} and K_{zx} are obtained from K_{xy} by cyclic permutations of x, y, and z. The initial data is evolved with unit lapse $\alpha = 1$ and vanishing shift $\beta^a = 0$. For the test results shown here, we use the values $A = 0.02$, $\varepsilon_1 = 0.01$, and $\varepsilon_2 = 0.0005$.

Figure 1 shows the common logarithm of the L_2 norm of the constraints as a function of time, with and without the off-shell modification terms in Eqs. (15). For the new constraint equations we have chosen

$$(\partial_\perp \mathcal{H})_{\text{new rhs}} = -0.4\mathcal{H} - \mathcal{L}_\beta \mathcal{H}, \tag{17a}$$

$$(\partial_\perp \mathcal{M}_a)_{\text{new rhs}} = -0.4\mathcal{M}_a - \mathcal{L}_\beta \mathcal{M}_a, \tag{17b}$$

so that at each point in space $\mathcal{H}$ and $\mathcal{M}_a$ have time

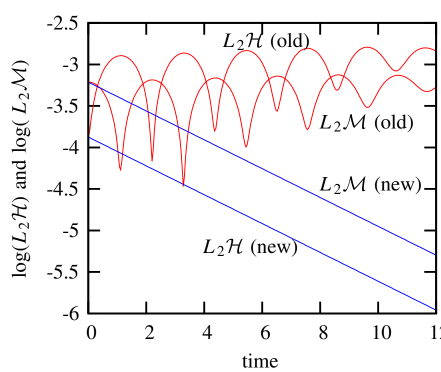

FIG. 1 (color online). Common logarithm of the L_2 norms of the Hamiltonian and momentum constraints, using the old and new equations of motion.

dependence $\sim \exp(-0.4t)$. The L_2 norms are defined without any factors of g_{ab} or $\sqrt{g}$. That is, we define $L_2\mathcal{H} = \sqrt{\sum \mathcal{H}^2/N^3}$ where the sum extends over the N^3 grid points. Similarly, we define $L_2\mathcal{M} = \sqrt{\sum \mathcal{M}_a\mathcal{M}_a/N^3}$ and include a sum over the index a. With these definitions, the only time dependence that should appear in the L_2 norms is the exponential decay $\sim \exp(-0.4t)$. This is precisely what we see in Fig. 1 with the new equations of motion. For the old equations of motion, we see strong oscillations on a short time scale and exponential growth on a longer time scale.

Our numerical code uses pseudospectral collocation [20] with a Fourier basis in each of the coordinate directions. Fourth-order Runge-Kutta is used for time stepping. The elliptic equations (14) are solved with the iterative method GMRES [21]. We use a left preconditioner consisting of the inverse of the diagonal part of the elliptic operator. One of the numerical issues that we face is spectral blocking [20]. This is the phenomenon in which aliasing causes an unphysical increase in power in the highest wave number modes that can be supported on the grid. Filtering can help alleviate this problem. For the simulations shown in Fig. 1, we use $N = 20$ collocation points in each dimension and a filter that sets the two highest frequencies to zero at the end of each time step. The time step is 0.04, compared to a light-crossing time of approximately 2π.

The results displayed in Fig. 1 show that we have indeed modified the equations of motion off-shell in such a way that unwanted growth in the constraints is eliminated. Ultimately, what we would like to show is the ability to prevent constraint growth in the first place. Our preliminary attempts to demonstrate this ability have not been completely successful, for reasons that we suspect are purely numerical. Although we cannot rule out the possibility that the combined system Eqs. (14) and (15) is mathematically ill-defined in some sense, the problems that we have encountered appear to be caused by numerical issues. One issue is spectral blocking, mentioned above. Another issue is the failure of our elliptic solver to converge to a solution under circumstances that we do not yet understand. We suspect that a better preconditioner will make our elliptic solver more robust and dependable.

For the simulation shown in Fig. 1, with the new equations of motion, the constraints continue to drop exponentially until $t \approx 15$. Beyond this time the constraints begin to suffer from high wave number variations whose growth counteracts the exponential drop of the longer wavelength modes. This breakdown is sensitive to the spatial resolution and amount of filtering and appears to be related to spectral blocking. With the old equations of motion, the Hamiltonian and momentum constraints continue to grow exponentially until $t \approx 45$. At that time $L_2\mathcal{H}$ has a value of $\sim 10^{-1}$ and the simulation breaks down. Again, this appears to be related to spectral blocking. We plan to study these issues in more detail and present further numerical tests in a future publication.

We would like to thank Ronald Fulp for helpful discussions. This work was supported by NASA Space Sciences Grant No. ATP02-0043-0056 and NSF Grant No. PHY-0600402.

[1] L. E. Kidder, M. A. Scheel, and S. A. Teukolsky, Phys. Rev. D **64**, 064017 (2001).

[2] L. Lindblom and M. A. Scheel, Phys. Rev. D **66**, 084014 (2002).

[3] H.-a. Shinkai and G. Yoneda, gr-qc/0209111.

[4] S. Detweiler, Phys. Rev. D **35**, 1095 (1987).

[5] M. Anderson and R. A. Matzner, Found. Phys. **35**, 1477 (2005).

[6] M. Tiglio, gr-qc/0304062.

[7] L. Lindblom, M. A. Scheel, L. E. Kidder, H. P. Pfeiffer, D. Shoemaker, and S. A. Teukolsky, Phys. Rev. D **69**, 124025 (2004).

[8] R. F. Stark and T. Piran, Comput. Phys. Rep. **5**, 221 (1987).

[9] M. W. Choptuik, Phys. Rev. Lett. **70**, 9 (1993).

[10] A. M. Abrahams and C. R. Evans, Phys. Rev. D **46**, R4117 (1992).

[11] A. M. Abrahams, G. B. Cook, S. L. Shapiro, and S. A. Teukolsky, Phys. Rev. D **49**, 5153 (1994).

[12] M. W. Choptuik, E. W. Hirschmann, S. L. Liebling, and F. Pretorius, Classical Quantum Gravity **20**, 1857 (2003).

[13] R. A. Matzner, Phys. Rev. D **71**, 024011 (2005).

[14] M. Holst, L. Lindblom, R. Owen, H. P. Pfeiffer, M. A. Scheel, and L. E. Kidder, Phys. Rev. D **70**, 084017 (2004).

[15] Y. Choquet-Bruhat, C. DeWitt-Morette, and M. Dillard-Bleick, *Analysis, Manifolds and Physics* (North-Holland, Amsterdam, 1977).

[16] J. W. York, in *Sources of Gravitational Radiation*, edited by L. Smarr (Cambridge University Press, Cambridge, England, 1979), pp. 83–126.

[17] G. B. Cook, Living Rev. Relativity **3**, 5 (2000).

[18] J. W. York, J. Math. Phys. (N.Y.) **14**, 456 (1973).

[19] J. D. Brown, Phys. Rev. D **71**, 104011 (2005).

[20] J. P. Boyd, *Chebyshev and Fourier Spectral Methods* (Dover, New York, 2001).

[21] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations* (SIAM, Philadelphia, 1995).

Chapter 9

Apparent Horizon Finder

The event horizon is commonly known as a location outside of a black hole from which light cannot escape gravity. More formally, it is the location in spacetime from which null geodesics cannot reach infinity. The analogue in 3+1 relativity is the apparent horizon, which is defined as the outermost marginally trapped surface. A marginally trapped surface is a surface in space from which null geodesics are not expanding [61, 62, 63].

9.1 Equations

In numerical relativity, there are two main motivations behind having an apparent horizon finder. The first motivation is to examine the properties of black holes. The apparent horizon gives important information about the black hole it contains, such as mass and angular momentum [64]. The other motivation is to define an excision boundary. Numerical simulations are not well suited to handle singularities such as the singular values inherent in a black hole. Singularities can be avoided through a method called 'excision'. When using excision, the area around the singular portion of a black hole is cut out, or excised. Information cannot travel from inside the event horizon to the surrounding spacetime, so it is theoretically acceptable to excise this region from the computational grid[1]. An apparent horizon is guaranteed to be on or within the event horizon, so it is physically appropriate to use the apparent horizon as an excision boundary. There is a third, less discussed use for an apparent horizon finder, and that is to probe the topology of spacetime [64].

A marginally trapped surface is a two-dimensional surface embedded in a spacelike hypersurface Σ with outward pointing normal s^i that satisfies the apparent horizon equation[2].

$$H = D_i s^i - K + s^i s^j K_{ij} = 0. \tag{9.1}$$

There may be many surfaces within Σ that satisfy $H = 0$ (marginally trapped surfaces), and the outermost marginally trapped surface is the apparent horizon. (It is also possible that the given space does not contain a horizon.) A surface in Σ may be parametrized by a function ϕ such that the surface is defined by

$$\phi(x^i) = 0 \tag{9.2}$$

where x^i are the spatial coordinates. A surface will be denoted as ϕ_S where $\phi_S \equiv \{x^i : \phi(x^i) = 0\}$. The outward pointing normal vector s^i to a surface can be written in terms of function ϕ as

$$s^i = g^{ij}\partial_j\phi / \sqrt{g^{kl}\partial_k\phi\partial_l\phi}. \tag{9.3}$$

This transforms the apparent horizon equation (9.1) into an equation in terms of the function ϕ. To solve the apparent horizon equation, we use a level set method combined with a curvature flow method[3].

Mean curvature flow is a method that evolves each point on a surface with a velocity dependent on the mean curvature at that point. The first term in the apparent horizon equation is equivalent to the mean curvature $\bar{H}$ of a 3D surface

$$\bar{H} = D_i s^i. \tag{9.4}$$

[1] Because the information in the interior of a horizon is irrelevant, it can be replaced by a nonsingular function to avoid singularities in a numerical simulation. This technique has been proposed and tested as an alternative to excision [65].

[2] As in chapter 2, Latin indices $i = 1, 2, 3$ will be used for spacelike quantities.

[3] There are many other ways to solve the apparent horizon equation, and a comprehensive overview is given in [62]

Surfaces that evolve with a velocity dependent on their mean curvature will evolve to a minimal surface or to a point [66]. This type of evolution is called 'mean curvature flow'. A surface moving under its mean curvature follows the equation

$$\frac{\partial x^i}{\partial \lambda} = -\bar{H} s^i \tag{9.5}$$

where λ is a pseudo-time parameter. If we want to look at the evolution of the function ϕ instead of x^i, we rewrite the equation making use of the chain rule[4] as

$$\frac{\partial \phi}{\partial \lambda} = \bar{H} \sqrt{g^{kl} \partial_k \phi \partial_l \phi}. \tag{9.6}$$

This is a parabolic equation for the unknown function ϕ that can be numerically evolved given an initial surface ϕ_λ at $\lambda = 0$. Starting with an initial surface ϕ_0, equation (9.6) flows each point on the surface ϕ_λ with a velocity $\bar{H}$ until $\bar{H}$ is zero. When $\bar{H}$ is zero, the surface stops evolving, indicating a minimal surface. The surface ϕ_∞ will be a minimal surface. Equation (9.6) is the mean curvature flow equation where $\bar{H}$ is the mean curvature. This is equivalent to the apparent horizon equation H only for time symmetric spatial hypersurfaces (i.e., when $K_{ij} = 0$). To find an apparent horizon in a general spatial hypersurface, we follow Tod [67] and assume that using a velocity function H will cause surfaces to flow to marginally trapped surfaces, i.e., that ϕ_∞ will be a marginally trapped surface. After replacing the mean curvature $\bar{H}$ with the apparent horizon equation H, our surface flow equation is now

$$\frac{\partial \phi}{\partial \lambda} = H \sqrt{g^{kl} \partial_k \phi \partial_l \phi}. \tag{9.7}$$

The trapped surface found may not be the outermost trapped surface (the apparent horizon). Starting with the largest possible ϕ_0 at the initial time will ensure that the trapped surface found is the apparent horizon, assuming the numerical domain is large enough to contain the apparent horizon.

Flowing a two-dimensional surface comes with a few drawbacks. The first is that one must start with a very large initial surface to ensure the minimal surface found is the outermost one. The second is that the surface cannot change topology. If

[4] See appendix A for an explanation.

the starting surface is a sphere, it cannot fission in order to find multiple minimal surfaces. To remedy these problems, we employ the level set method developed by Sethian and Osher [68]. The method entails flowing a higher-dimensional surface Φ_λ which does not change topology. The surfaces ϕ_λ within Φ_λ can change topology.

As an example of a level set and to define some terminology, consider flowing the surface of a sphere of radius 2 according to the speed function of the apparent horizon equation. If the surface flows according to a speed function H that happens to contain two disjoint apparent horizons, we will have numerical problems as the surface becomes highly distorted around the horizons. Numerically, the surface cannot bifurcate to indicate the two separate horizons. To avoid this problem, we use the level set method. We choose a 'level set function' that contains our surface, in this case, the surface of a sphere of radius 2.

For this example, we choose our level set function as $\phi(x^i)$ where

$$\phi(x^i) \equiv x^2 + y^2 + z^2 - 4. \tag{9.8}$$

The function ϕ is defined at all points in three-dimensional space. We can define two-dimensional surfaces within that space by setting $\phi(x^i) = c$ where c is a constant. Varying c gives various surfaces that we will call ϕ_c making up what is called a 'level set of surfaces'. In this example, varying c gives a level set of spherical surfaces having radius $\sqrt{c+4}$. The zero level set $\phi_{c=0}$ is the set of all points such that $c = 0$, in this case a spherical shell of radius 2. (Usually we choose our surface of interest to be the zero level set of a level set function.) The equation $\phi = c$ defines a two surface in $I\!R^3$. Flowing a two surface according to the flow equation (9.7) can create problems as mentioned previously. The level set method by Sethian and Osher [68] suggests flowing a higher-dimensional surface. Instead of a two surface in $I\!R^3$ we now consider a three surface in $I\!R^3 \times [0, \infty)$ (see [69, 70]). Our higher-dimensional level set function is Φ and our higher-dimensional surface is defined by $\Phi(x^i, \lambda) = C$.

Using this approach, our flow equation is now

$$\frac{\partial \Phi}{\partial \lambda} = H \sqrt{g^{kl} \partial_k \Phi \partial_l \Phi} \tag{9.9}$$

where H is the apparent horizon equation (9.1) and

$$s^i = g^{ij}\partial_j\Phi/\sqrt{g^{kl}\partial_k\Phi\partial_l\Phi}. \tag{9.10}$$

The original curvature flow equations started with an initial two surface ϕ_0 and evolved it as a parabolic flow equation. The original initial surface ϕ_0 is defined as

$$\phi_0 = \{x^i \in I\!R^3 : \phi(x^i) = 0\}. \tag{9.11}$$

Similarly, we need to specify an initial three surface to evolve using equation (9.9). We will choose as our initial three surface

$$\Phi(x^i, \lambda = 0) = \phi(x^i). \tag{9.12}$$

All of the surfaces $\phi(x^i) = c$ within Φ are also evolving according to their mean curvature, but because Φ is the function being evolved there are no numerical problems when the surfaces $\phi(x^i) = c$ change topology—the function Φ does not change topology. To recover the new location of the original surface of interest at a time $\lambda = \Lambda$, we look at $\Phi(x^i, \Lambda) = 0$. Recall the original surface of interest defined as $\phi(x^i) = 0$ belongs to a level set of surfaces $\phi(x^i) = c$. In most cases, the surface of interest is chosen as the zero level set of ϕ, but this is not necessary. For example, one could instead track the surfaces defined by $\phi(x^i) = 3$. If we evolve Φ using equation (9.9), we are simultaneously flowing a continuum of all surfaces ϕ_c within Φ. Tracking a certain surface ϕ_c translates into looking at isosurfaces of the function Φ with a value of c. All surfaces will flow toward the trapped surfaces and then stop evolving. The configuration of the surfaces when they stop evolving indicate the trapped surfaces, the outermost surface being the apparent horizon.

When evolving Φ, surfaces that satisfy $H = 0$ will have a 'speed' of 0. All of the surfaces ϕ_c in the level set function Φ are flowing in pseudo-time. (From here on, pseudo-time parameter λ will be relabeled as t.) Instead of tracking a particular surface ϕ_c until it stops evolving, we can look at the values of Φ at points x^i at different numerical timesteps n, or $\Phi(x^i, t_n) - \Phi(x^i, t_{n-1})$. The set of points that are not changing, $\{x^i : \Phi(x^i, t_n) - \Phi(x^i, t_{n-1}) = 0\}$, indicate where the surfaces are not evolving. This is demonstrated further in section 9.2.

A third drawback to flowing a two-dimensional surface that was not mentioned previously is that the numerical integration can be very slow due to the necessarily small timesteps indicated by the Courant condition. In addition, the flow moves according to a speed function that approaches zero, also making things slow. The adding of an extra dimension makes this method even slower, but in return we can ensure that multiple minimal surfaces of any topology may be found. The speed (or lack thereof) makes this method unsuitable for the use of dynamical horizon tracking during a numerical evolution, but speed is of lesser importance when it is only necessary to find horizons once on a single spacial hypersurface. Most apparent horizon finders currently in use assume prior knowledge of the apparent horizon topology. Although a single black hole or close binary black hole horizon is thought to have a trivial topology, there remains the possibility for multiple horizons or toroidal horizons. There are also apparent horizons of non-trivial topology expected in the study of black strings and other exotic matter [71, 62, 72, 73]. Our horizon finder would be of the greatest use in these cases.

There are two other implementations of apparent horizon finders that make use of a mixed level set and curvature flow idea. The 'Level Flow' apparent horizon finder in [74, 75] flows a two-dimensional surface. In level flow, the level set idea is not applied to the function $\psi(x^i) = 0$ but to the apparent horizon equation $H = 0$. H is utilized as a level set function $H = c$ where c is a constant and $H = 0$ is the zero level set which recovers the apparent horizon equation. The only other implementation of an apparent horizon finder using the level set method of higher-dimensional surface flow is [70], and it is only for time symmetric spatial metrics.

9.2 Code Tests

In this section, we will describe multiple numerical experiments that we use to test the performance of our apparent horizon finder.

9.2.1 Flat Space Tests

Any smooth, closed two-dimensional surface in Euclidean space moving under its mean curvature will evolve to a minimal surface or to a point [66]. Figure 9.1 shows an initial surface of a sphere evolving under its mean curvature in flat space. We choose to track as our starting surface a sphere with radius 9. Our initial Φ is a function such that a sphere with radius 9 has values $\Phi = 0$:

$$\Phi = \sqrt{x^2 + y^2 + z^2} - 9. \tag{9.13}$$

The plot shows the zero level contours of Φ at various t in the z=0 plane (our code is three-dimensional). The outermost circle is the location of the surface at the initial time, and the inner contours are at future times until the surface disappears. Figure 9.2 shows as a starting surface an ellipsoid evolving under its mean curvature. As expected, it evolves into a sphere and then into a point (disappears). Note here that if we were to look at $\Phi_t(x^i) - \Phi_{t-1}(x^i)$ and plot the zeros in order to find where the surfaces are not evolving, there would be nothing on the graph. The curves are always changing—the minimal surface is a point. The appearance of a squarish shape in the figures is an artifact from plotting guard cells. There are extra points output to the data file (because the guard cells overlap). The extra points along the square shaped grids make the points along the grid borders darker, giving the appearance of a square in the middle of the contour plots.

9.2.2 Misner Data

There are various initial data configurations that are proposed as test beds for apparent horizon finders [61]. One such test is 'Misner Data' [76]. Misner data is initial data for time symmetric, axisymmetric, equal mass black holes. The metric for Misner data is conformally flat

$$ds^2 = \psi^4(dx^2 + dy^2 + dz^2) \tag{9.14}$$

with a conformal factor ψ as

$$\psi = 1 + \sum_{n=1}^{N} \frac{1}{\sinh(n\mu)} \left(\frac{1}{{}^{+}r_n} + \frac{1}{{}^{-}r_n} \right) \tag{9.15}$$

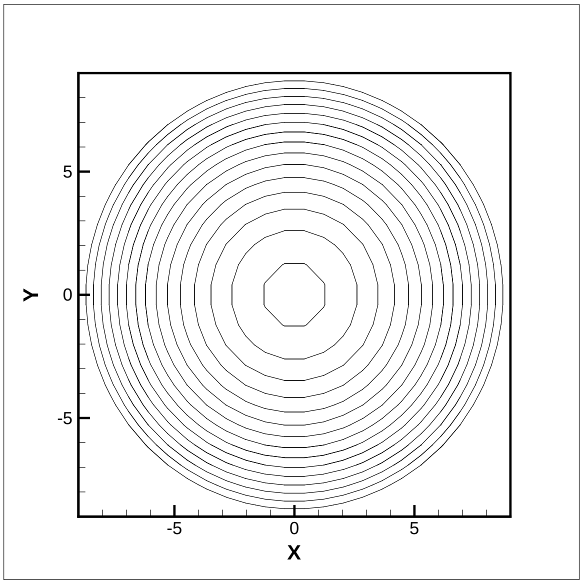

Figure 9.1: A spherical surface evolves under its mean curvature in Euclidean space. The outermost curve is at the initial time, and the inner curves are at future times. The surface shrinks, then disappears.

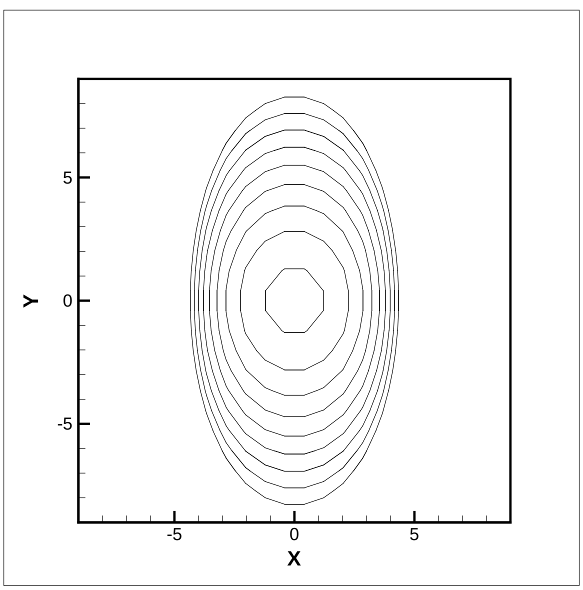

Figure 9.2: The surface of an ellipsoid evolves under its mean curvature in Euclidean space. The outermost curve is at the initial time, and the inner curves are at future times. The surface deforms to a sphere, shrinks, then disappears.

where $N = \infty$ (for our computations, we use $N = 100$), and

$$\pm r_n = \sqrt{\left(x \pm \frac{1}{\tanh(n\mu)}\right)^2 + y^2 + z^2}. \tag{9.16}$$

Misner data gives a set of two black hole initial data parametrized by the distance μ between holes, with a single horizon bifurcating at about $\mu = 1.36$ [61]. We test two cases of Misner data here. All plots are constant value contours in the $z = 0$ plane. The first case is Misner data with parameter $\mu = 1.8$. Figure 9.3 shows a contour plot of the zeros of $\Phi_t - \Phi_{t-1}$. The next plot (figure 9.4) follows the evolution of the surface $\Phi_t = -7.5$. It shows the surface, initially a sphere, evolving to be smaller and then splitting into two distinct horizons. (From now on, we will refer to all trapped surfaces as 'horizons'. In the case that there are two or more trapped surfaces, they will be referred to as 'inner' or 'outer' horizons with the understanding that the outermost horizon is the apparent horizon.) Figure 9.5 shows an evolution sequence starting from an initial surface that did not enclose the horizons ($\Phi_t = -7.98$). It shrinks and then expands into the horizons. Note that the evolution of data set $\mu = 1.8$ was carried out only once; figures of the evolution of different initial surfaces are made by choosing different isosurfaces in Φ. Figure 9.6 shows the zeros of $\Phi_t - \Phi_{t-1}$ for Misner data with $\mu = 1.0$.

9.2.3 Brill Wave Test

Another standard test case is a Brill wave [45]. Brill waves are strong gravitational waves. High amplitude Brill wave initial data may contain an apparent horizon. The Brill metric is

$$ds^2 = \psi^4[e^{2q}(d\rho^2 + dz^2) + \rho^2 d\phi^2], \tag{9.17}$$

where we choose the function q to be of the form [77]

$$q = a\rho^2 e^{-r^2}. \tag{9.18}$$

In terms of Cartesian coordinates, in the above equations $\rho^2 = x^2 + y^2$ and $r^2 = x^2 + y^2 + z^2$. The amplitude of the Brill wave is given by a. We solve for initial

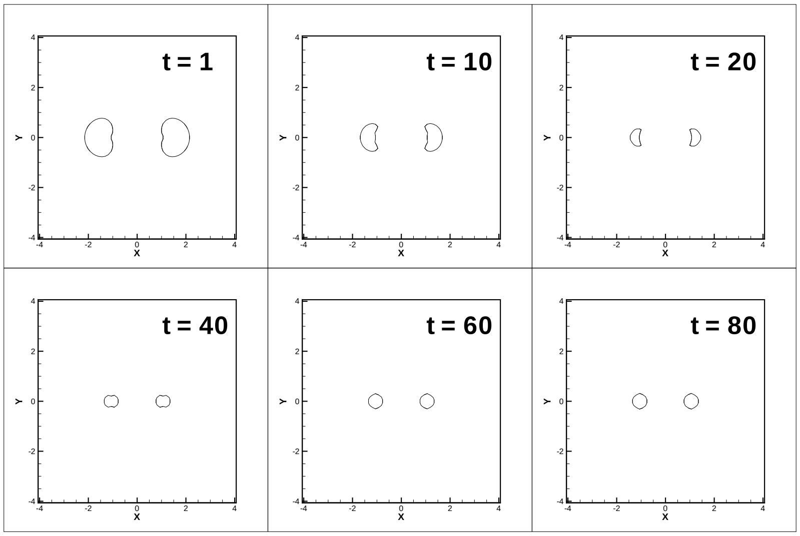

Figure 9.3: Evolution of the zero level contours of $\Phi_t - \Phi_{t-1}$ for two black hole Misner data with separation parameter $\mu = 1.8$. The final two panels show the approximate locations of the apparent horizons.

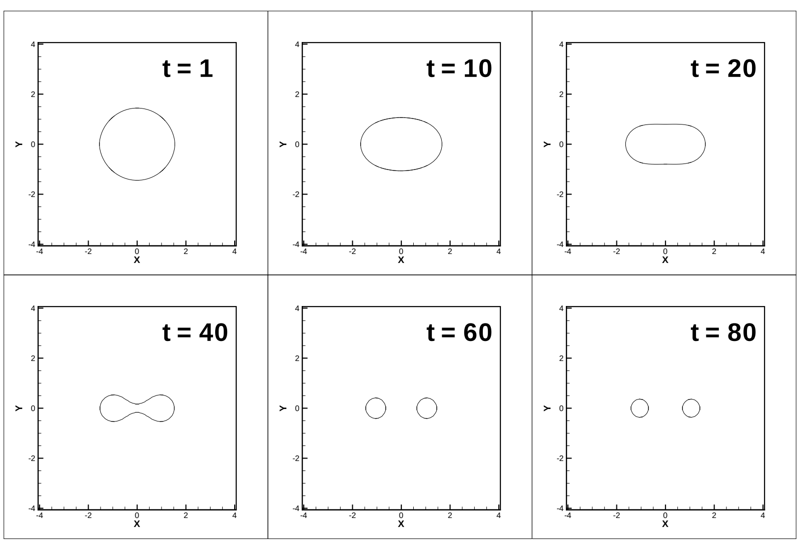

Figure 9.4: Evolution of the $\Phi = -7.5$ surface to the approximate location of the apparent horizons for Misner data with separation parameter $\mu = 1.8$. The original surface $\Phi = -7.5$ is a sphere that encloses the horizons. The surface shrinks, then bifurcates in order to flow to the horizons.

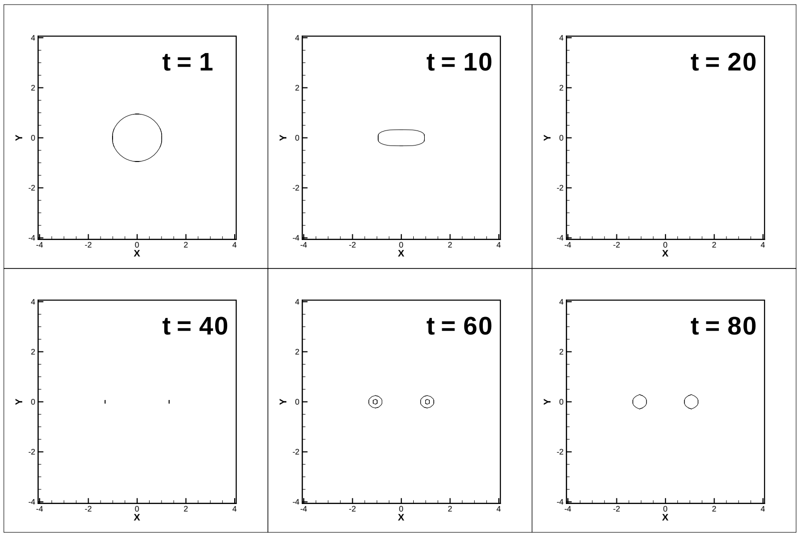

Figure 9.5: Evolution of the $\Phi = -7.98$ surface to the approximate location of the apparent horizons for Misner data with separation parameter $\mu = 1.8$. The original surface $\Phi = -7.98$ is a sphere that does not enclose the horizons. The surface shrinks, then expands into the apparent horizons.

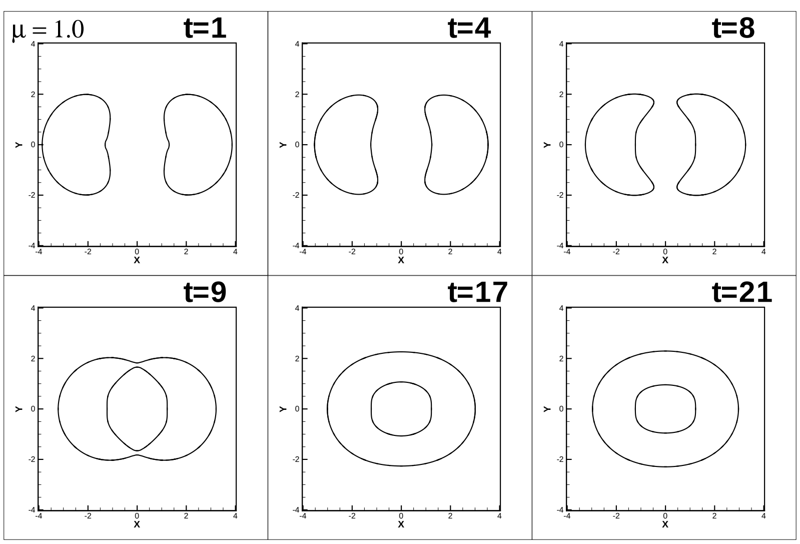

Figure 9.6: Evolution of the zero level contours of $\Phi_t - \Phi_{t-1}$ for two black hole Misner data with separation parameter $\mu = 1$. The final panels shows the approximate locations of an inner horizon and the apparent horizons.

data by choosing a moment of time symmetry and solve the resulting Hamiltonian constraint equation

$$H = \nabla^2\psi + \frac{1}{4}\left(\frac{\partial^2 q}{\partial\rho^2} + \frac{\partial^2 q}{\partial z^2}\right)\psi = 0. \tag{9.19}$$

In equation (9.19), ∇^2 is the Laplacian in flat space. Figures 9.7 and 9.8 show the zeros of $\Phi_t - \Phi_{t-1}$ for Brill wave initial data with amplitude $a = 12$. The conformal factor ψ was calculated by AMRMG using adaptive mesh refinement. Figure 9.9 shows the grid structure along with the zero contours of $\Phi_t - \Phi_{t-1}$ for the latest calculated time step. Octant symmetry is used and the grid spans $[0, 8]^3$. In the figures that show the grid structure, each box contains 8^2 points (there are 8^3 points per block). The innermost contours appear to be breaking up around the refinement borders. We perform a second evolution using the same parameters with no refinement (unigrid) and a smaller box, $[0, 4]^3$. There are 8^3 boxes with 8^3 grid points each giving a total of 64^3 grid points and a resolution of 0.0625. Figure 9.10 shows the zeros of $\Phi_t - \Phi_{t-1}$ in the three planes. The two outermost surfaces found are visually compatible with the inner and outer horizons published in the study of apparent horizon finder testbeds [61]. There are still odd contours in the center of the grid. Figure 9.12 shows a contour plot of Φ, with contour levels from -10.5 to 3.5, by .25. Figure 9.12 shows the same at the last calculated timestep. The contours are not breaking up, just the differences in contours. It may be that the breaking up of contours for $\Phi_t - \Phi_{t-1}$ is an artifact of interpolation (on to a plane) and subtraction, and that the odd contours shown are from not having high enough resolution. Also, as time evolves, each surface in the continuum will flow toward the location of the horizon and then stop. The 'piling up' of surfaces will likely create odd numerical behavior. Tracing a single contour closest to the horizon should give the clearest indication of the horizon location. We cannot look at contour evolution for the AMR run, as at the time of those runs, Φ_t data was not saved, only $\Phi_t - \Phi_{t-1}$. Another run using AMR and tracing a Φ constant contour is necessary for further investigation.

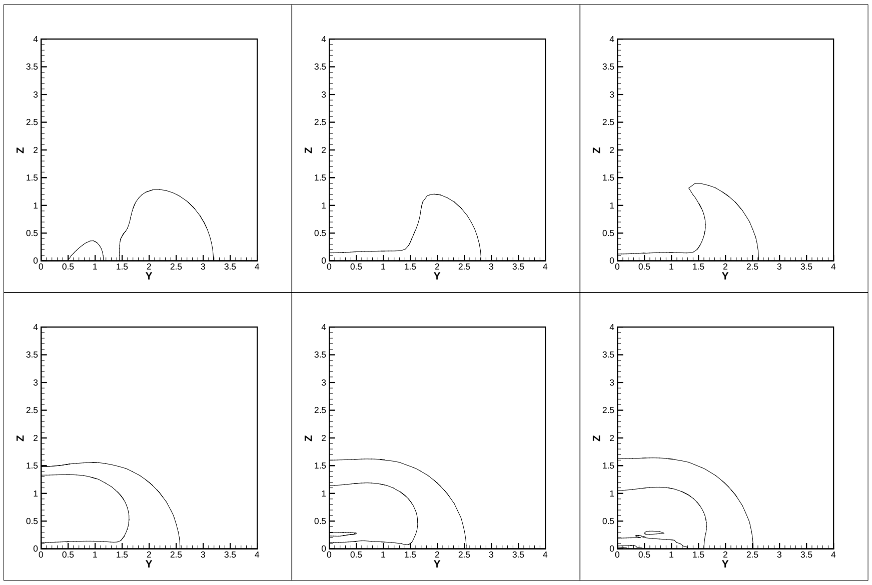

Figure 9.7: Early stages of an evolution of the zeros of the function $\Phi_t - \Phi_{t-1}$ in Y-Z plane for Brill wave initial data with $a = 12$. Last panel shows the outer contours converging to the published location of the apparent horizon, while inner contours are still evolving.

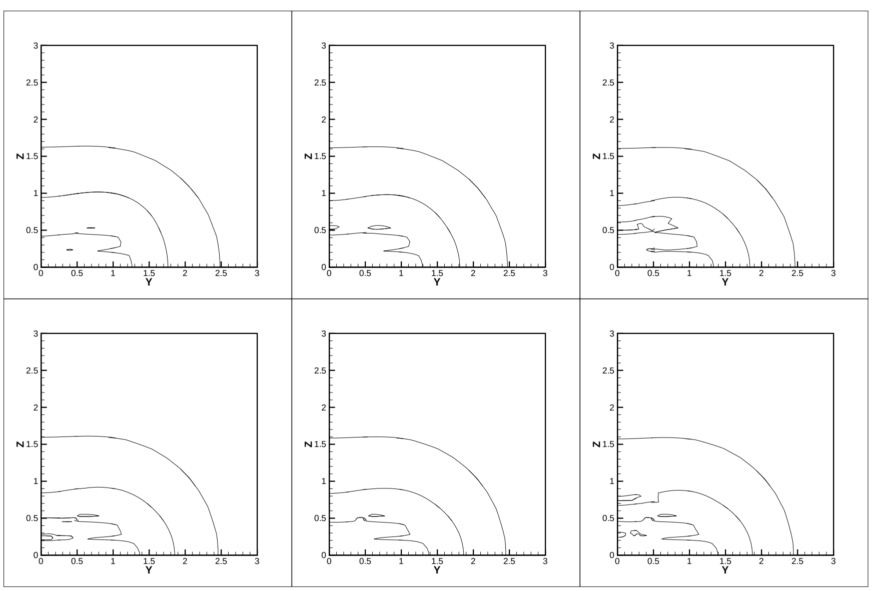

Figure 9.8: Later stages of an evolution of the zeros of the function $\Phi_t - \Phi_{t-1}$ in Y-Z plane for Brill wave initial data with $a = 12$. Last panel shows the outer contours shrinking past the published location of the apparent horizon. The inner contours evolve to the published location of the inner horizon then begin breaking up.

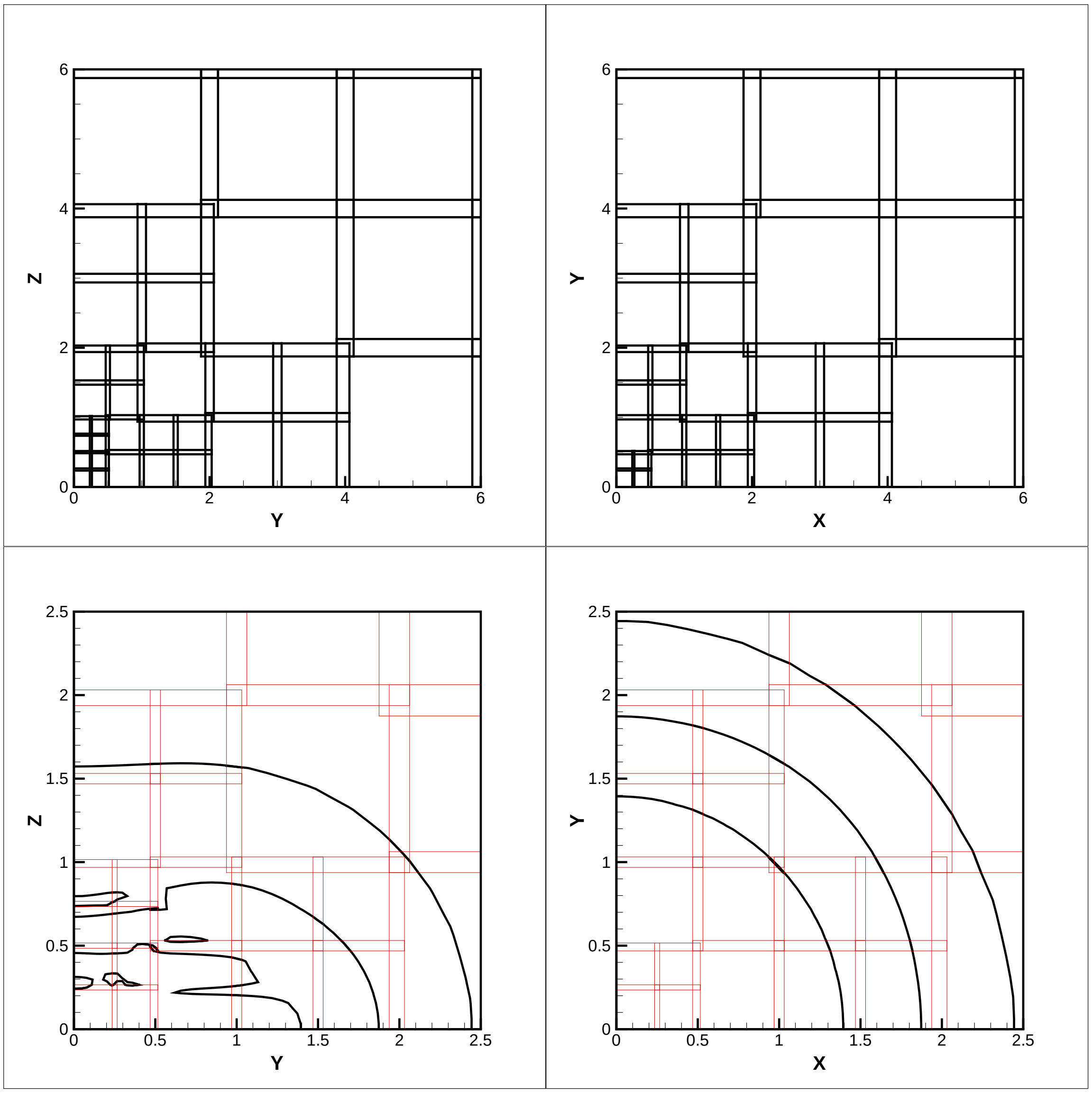

Figure 9.9: The top panels show the AMR grid block boundaries for the Brill wave $a = 12$ test case for the entire domain [0,6]. The bottom two panels show the zeros of function $\Phi_t - \Phi_{t-1}$ in Z and X planes along with the grid structure for the domain [0,2.5]. The bottom left panel suggests that the contours are breaking up along grid refinement boundaries.

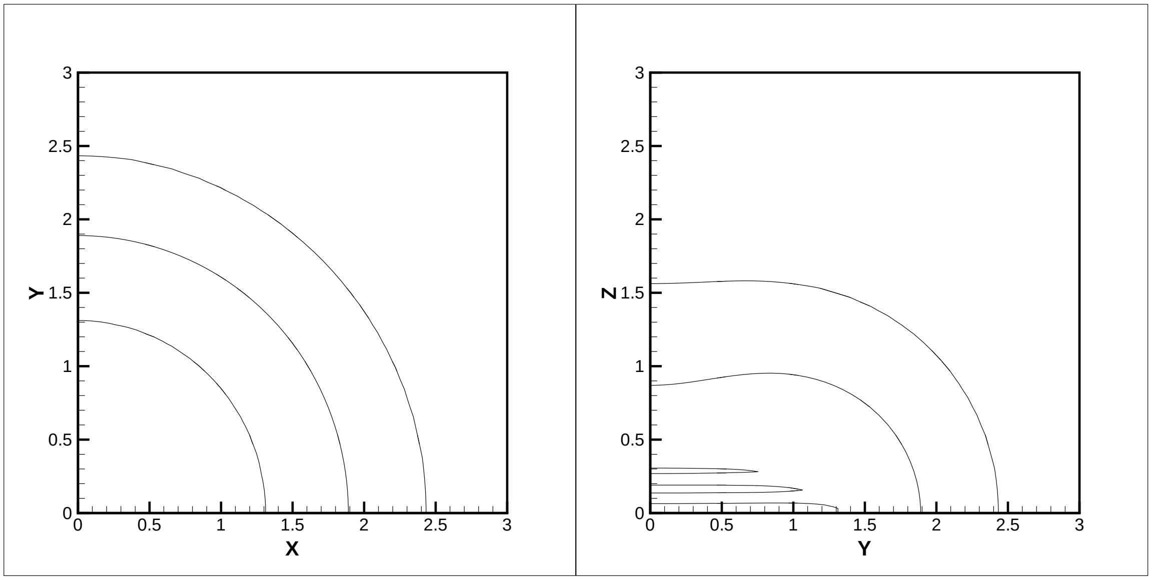

Figure 9.10: Zeros of the function $\Phi_t - \Phi_{t-1}$ at mid evolution using unigrid for a Brill wave metric with $a = 12$.

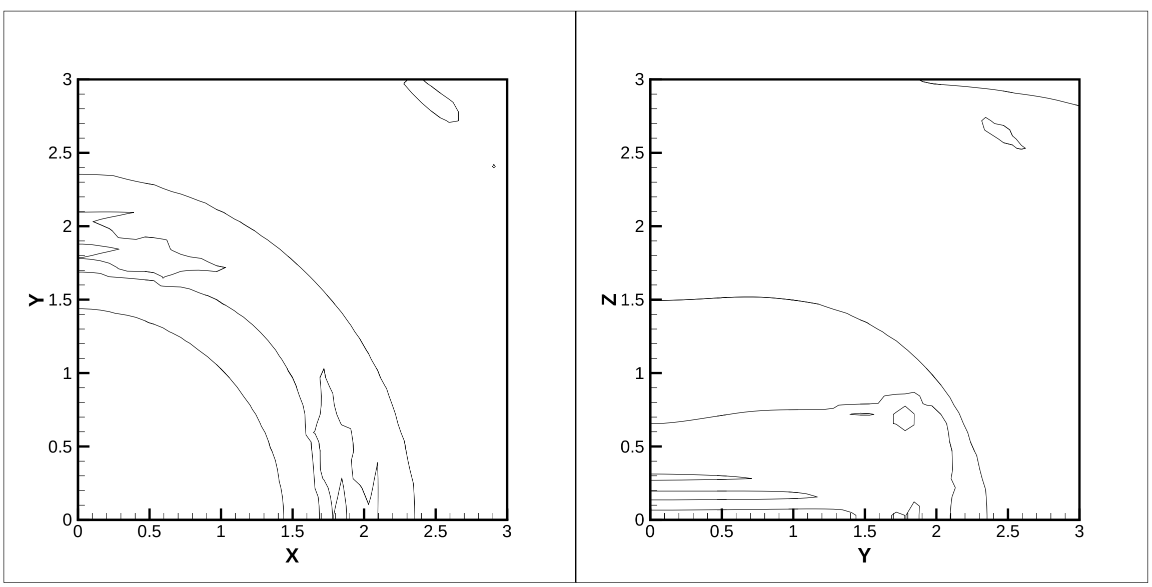

Figure 9.11: Zeros of the function $\Phi_t - \Phi_{t-1}$ at final calculated time using unigrid for a Brill wave metric with $a = 12$.

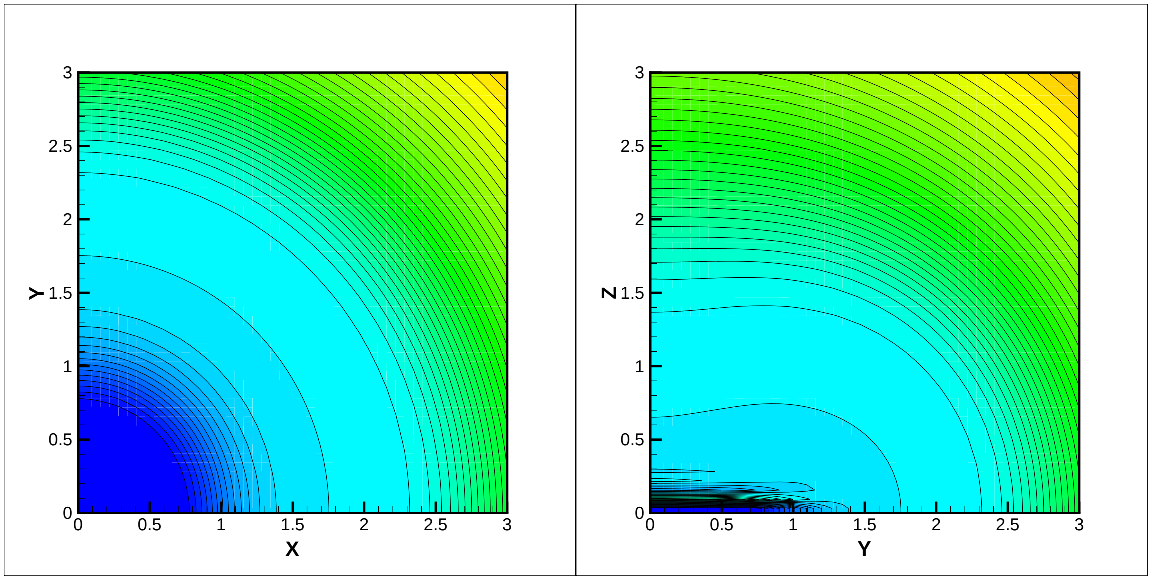

Figure 9.12: Contour levels of function Φ at mid evolution time using unigrid for a Brill wave metric with $a = 12$. The original surfaces at $t = 0$ were spherical and equidistant.

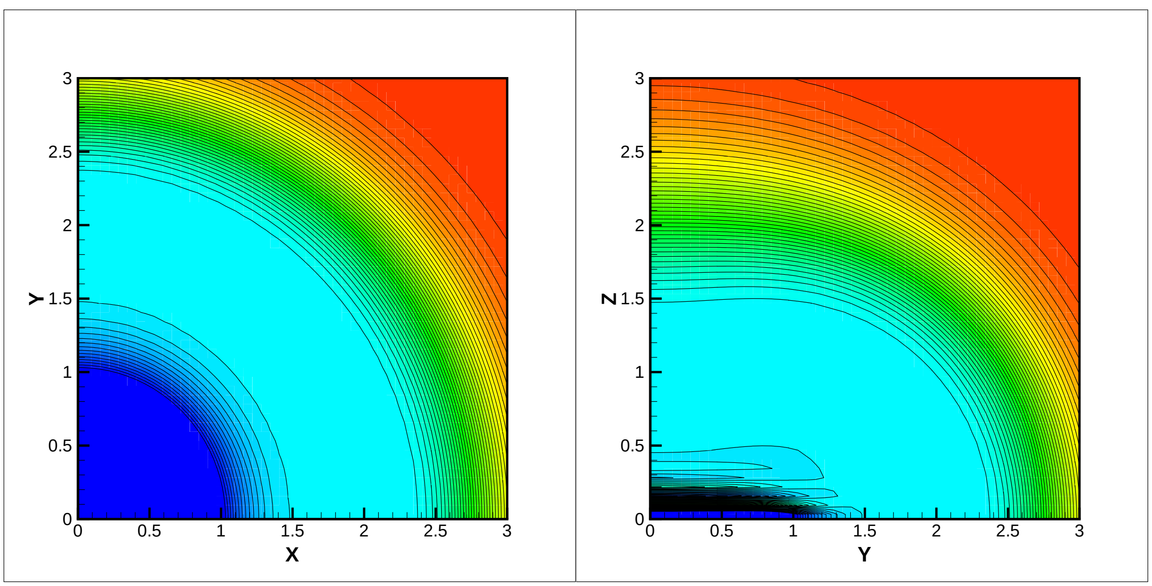

Figure 9.13: Contour levels of function Φ at the final calculated time using unigrid for a Brill wave metric with $a = 12$. The original surfaces at $t = 0$ were spherical and equidistant.

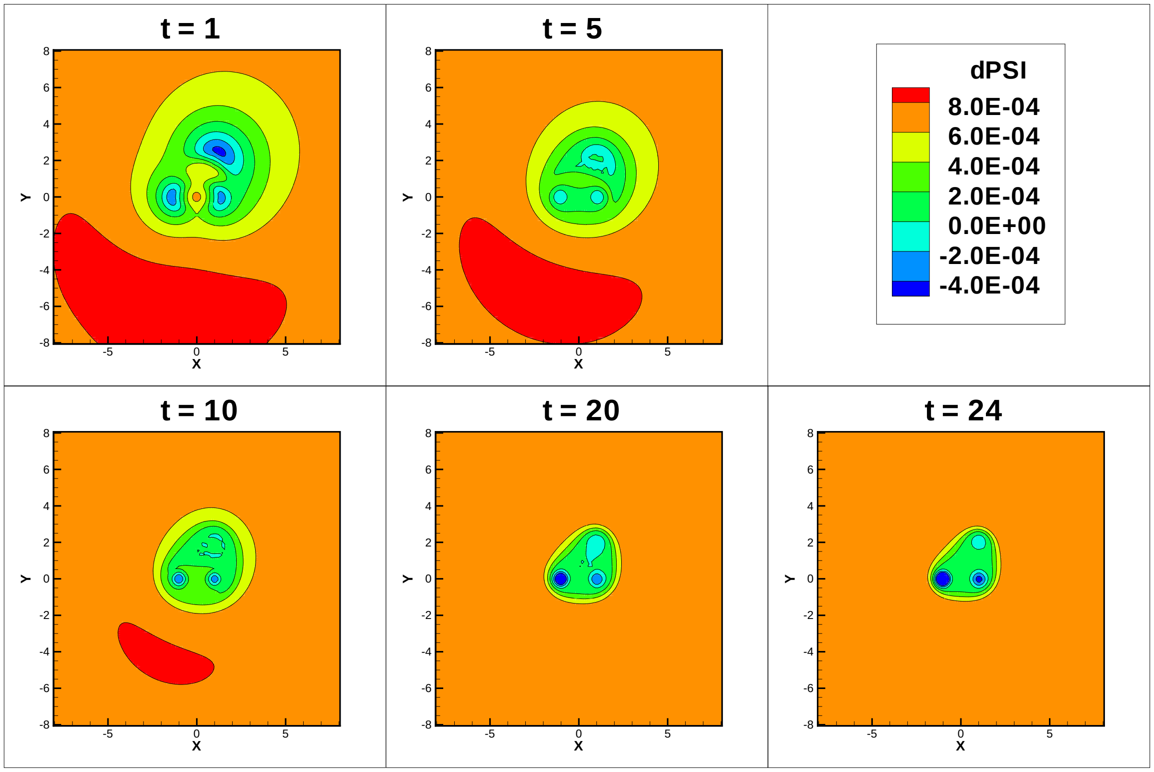

Figure 9.14: This figure shows the evolution of function $\Phi_t - \Phi_{t-1}$ in a three Schwarzschild black hole metric.

9.2.4 Multiple Black Holes in Time Symmetry

Our apparent horizon finder can handle any number of black holes. In figure 9.14 and figure 9.15, we show data for three equal mass Schwarzschild black holes. We start with a conformally flat metric and add three black holes by choosing a conformal factor

$$\psi = 1 + \sum_{n=1}^{N} \frac{m_{(n)}}{2|\vec{r} - \vec{r}_{(n)}|} \tag{9.20}$$

where there are $N = 3$ black holes each having mass $m_{(n)} = 1$ with locations $r_{(n)}$ of the n^{th} black hole at (-1,0,0), (1,0,0), and (1,2,0).

Figure 9.14 shows a contour plot of $\Phi_t - \Phi_{t-1}$, labeled ‘dPsi’ in the contour legend. Figure 9.15 shows the zero level contours of $\Phi_t - \Phi_{t-1}$. Figure 9.16 shows the evolution of surface $\Phi = 3.325$. The grid domain is $[-8, 8]^3$ with 128^3 grid points giving a resolution of 0.125.

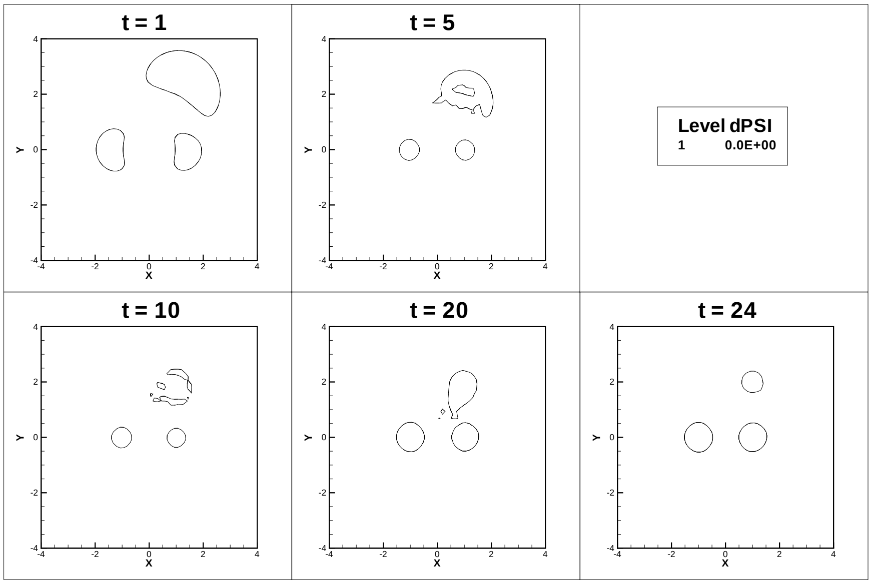

Figure 9.15: The plots show the evolution of the zero level surfaces of the function $\Phi_t - \Phi_{t-1}$ in a three Schwarzschild black hole metric. The last panel shows the approximate locations of black hole horizons.

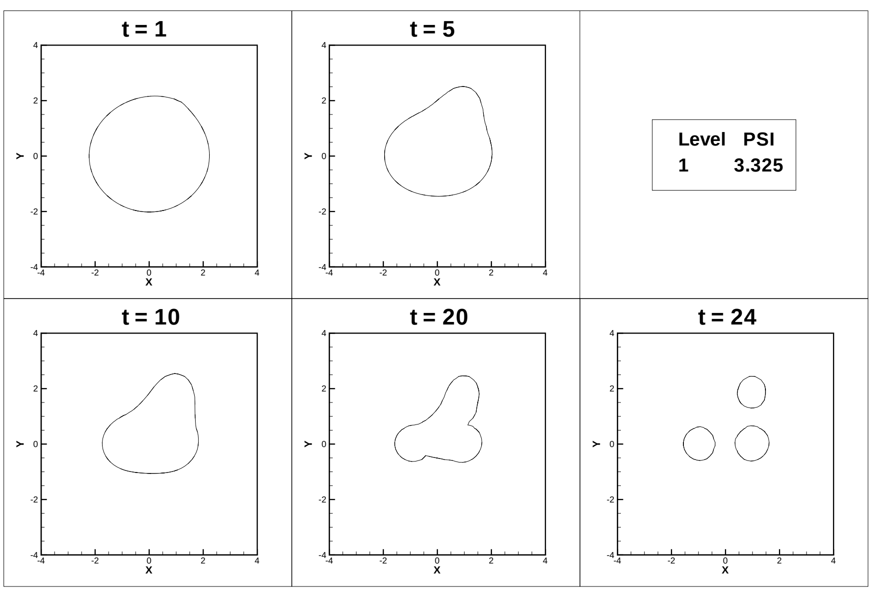

Figure 9.16: The plots show the evolution of the surface $\Phi = 3.325$ in a three Schwarzschild black hole metric. The last panel shows the approximate locations of the black hole horizons.

9.2.5 Multiple Black Holes without Time Symmetry

The only other implementation of a higher-dimensional level set apparent horizon finder (that we know of) is Pasch [70], and it is only for time symmetric spaces. We present two examples of apparent horizons as test cases for our AH finder with nonzero K_{ij} using puncture data with mass, momenta and spin. We use the Bowen–York analytical representation for a non-time-symmetric black hole. The trace of K_{ij} is chosen so that $K = 0$, and momentum constraint is solved analytically by choosing [78]

$$\begin{aligned}\tilde{K}^{ij} = \; & \tfrac{3}{2r^2}(P^i n^j + P^j n^i - (\tilde{g}^{ij} - n^i n^j)P^k n_k) \\ & + \tfrac{3}{r^3}\left(\epsilon^{ikl} S_k n_l n^b + \epsilon^{jkl} S_k n_l n^i\right) . \end{aligned} \tag{9.21}$$

In equation (9.21), $\tilde{K}_{ij}$ is the conformal extrinsic curvature such that $\tilde{K}_{ij} = \psi^2 K_{ij}$, $\tilde{g}_{ij}$ is the conformal metric such that $\tilde{g}_{ij} = \psi^{-4} g_{ij}$, n^i is the unit radial vector x^i/r, P^i is the momentum vector, S^i is the spin vector, and ϵ^{ijk} is the Levi–Civita symbol[5]. Note that the conformal metric $\tilde{g}_{ij}$ is flat, so tensor indices on conformal variables can be raised and lowered without adding factors of the conformal factor ψ. Equation (9.21) can be generalized to represent multiple black holes by using

$$\tilde{K}^{ij} = \sum_{n=1}^{N} \tilde{K}^{ij}_{(n)} \tag{9.22}$$

where N is the number of black holes and $\tilde{K}^{ij}_{(n)}$ is the conformal extrinsic curvature for the n^{th} black hole as given by equation (9.21) We solve the Hamiltonian constraint for the nonsingular part of the conformal factor ψ using the puncture method as described in section 3.3. The Hamiltonian constraint is given by

$$\nabla^2 u + \beta(1 + \alpha u)^{-7} = 0 \;, \tag{9.23}$$

where

$$\beta = \frac{1}{8}\alpha^7 \tilde{K}^{ij}\tilde{K}_{ij} \quad \text{and} \quad \alpha = 1/\sum_{n=1}^{N} \frac{m_{(n)}}{2|\vec{r} - \vec{r}_{(n)}|} \tag{9.24}$$

[5] The Levi–Civita symbol is $\epsilon^{ijk} = 1$ for even permutations of $(1, 2, 3)$, $\epsilon^{ijk} = -1$ for odd permutations of $(1, 2, 3)$, and $\epsilon^{ijk} = 0$ if there are repeated indices.

and u is the nonsingular part of the conformal factor ψ:

$$\psi = u + \sum_{n=1}^{N} \frac{m_{(n)}}{2|\vec{r} - \vec{r}_{(n)}|}. \tag{9.25}$$

The mass of the n^{th} black hole is given by $m_{(n)}$, $\vec{r}$ is the radial coordinate vector, and $\vec{r}_{(n)}$ is the location of the n^{th} black hole.

The testbed paper [61] has an example of finding the apparent horizon for two black holes. We use this to test our finder for the non-time-symmetric case. The data is for two black holes with the following parameters: Black hole $n = 1$ has mass $m = 1.5$, momentum $P^i = (2, 0, 0)$, spin $S^i = (-0.5, 0.5, 0)$ and location $(0, 0, 0.5)$. Black hole $n = 2$ has mass $m = 1$, momentum $P^i = (-2, 0, 0)$, spin $S^i = (0, 1, 1)$ and location $(0, 0, -0.5)$. Figure 9.17 shows contour plots of u at contour levels 1.05 to 1.11 by 0.01 and figure 9.18 shows the horizons found via the flow of the $\Phi = -1.8$ contour level as to compare with figure 15 in [61]. The horizon locations are similar to what is in their paper, but not the same. In particular, our Z axis horizon appears smaller, and our X axis horizon does not appear to lean as much to the left as theirs[6]. The test should be rerun with exactly the same parameters. To save computational time and achieve higher resolution while still using unigrid, we use a smaller grid domain $[-3, 3]^2$. Figure 9.22 shows how the $\Phi_t - \Phi_{t-1}$ contours break up at late times. The time shown is the same as in figure 9.18; the Φ =constant contours are not affected. The break up seems to be an artifact of the subtraction. More work needs to be done to determine the best way of analyzing code output.

Next we explore an initial data set with three black holes using AMR. The data for the three black holes have the following parameters: Black hole $n = 1$ has mass $m = 1$, momentum $P^i = (-1, 1, 0)$, spin $S^i = (0, 0, 12)$ and location $(0, 0, 0)$. Black hole $n = 2$ has mass $m = 1$, momentum $P^i = (1, -1, 0)$, spin $S^i = (0, 0, -12)$ and location $(-2, -3, 0)$. Black hole $n = 3$ has mass $m = 1$, momentum $P^i = (0, 0, 1)$, spin $S^i = (12, 0, 0)$ and location $(2, 2, 0)$. Figure 9.20 shows the function u in the X, Y, and Z planes for the contour levels $u = \{1.2, 1.3, 1.4, 1.55, 1.7, 2.0, 2.7\}$. Figure

[6] It is difficult to tell from inspection how different the horizons are. It would be useful to request the data files for the testbed paper horizons to do a direct comparison.

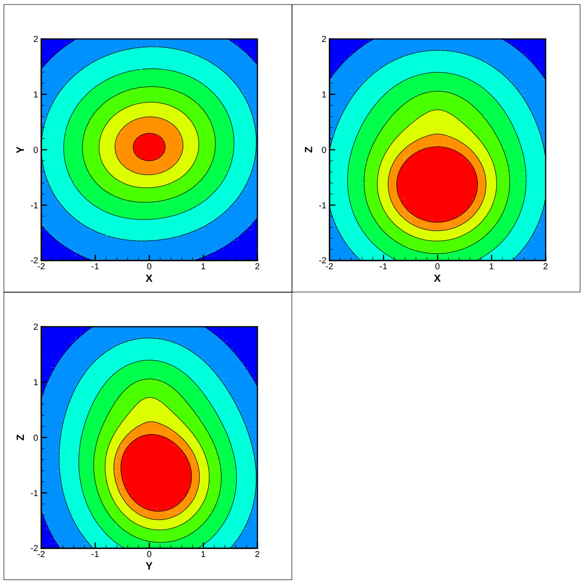

Figure 9.17: Non-singular part (u) of conformal factor ψ for the non-time-symmetric two black hole test case for the contour levels (outermost to innermost) $u = \{1.05, 1.06, 1.07, 1.08, 1.09, 1.10, 1.11\}$.

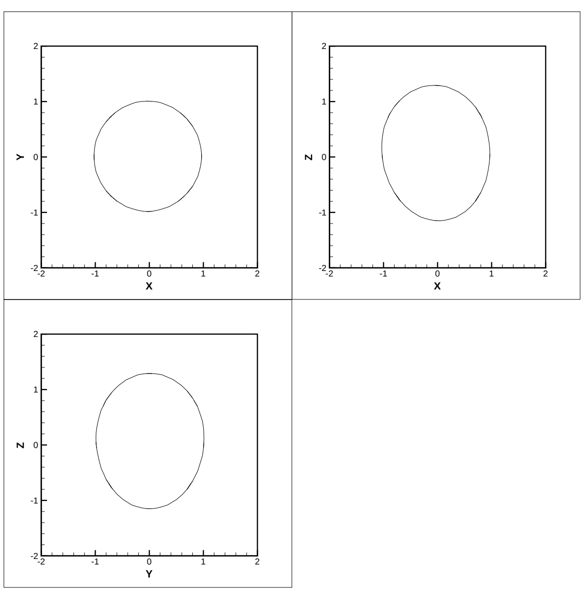

Figure 9.18: Horizon location for the non-time-symmetric two black hole test case.

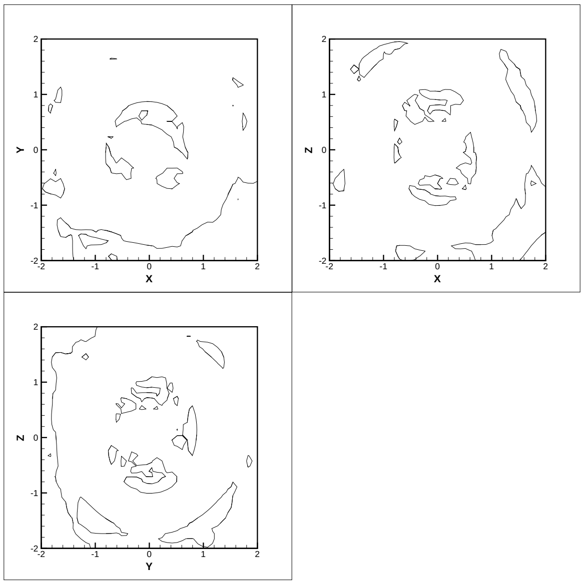

Figure 9.19: Zeros of function $\Phi_t - \Phi_{t-1}$ for the non-time-symmetric two black hole test case at late times.

9.21 shows the mesh as chosen by AMRMG. Figure 9.22 and 9.23 show the zeros of $\Phi_t - \Phi_{t-1}$ and the evolution of two surfaces $\Phi = -6$ and $\Phi = -5.37$.

9.3 Code Details

The AHF code needs as initial input the spatial metric g_{ij}, the spatial extrinsic curvature K_{ij} and an initial guess for $\Phi(x, y, z, t = 0)$. The metric and curvature can be set analytically or they can be set from a numerically generated solution to the constraint equations via a checkpoint file created by AMRMG.

Level set methods can have some stability problems. We have found that the Crank Nicholson routine that is generally used was not stable for long times. The integration routine that we use is Euler with the 'back and forth error compensation and correction' scheme as suggested by [79]. The forward backward error correction scheme increases the order of a scheme by one, so our integrator is second order accurate in time. The spatial finite differencing used is also second order. We use PARAMESH for parallelism and grid structure. PARAMESH provides a flux matching

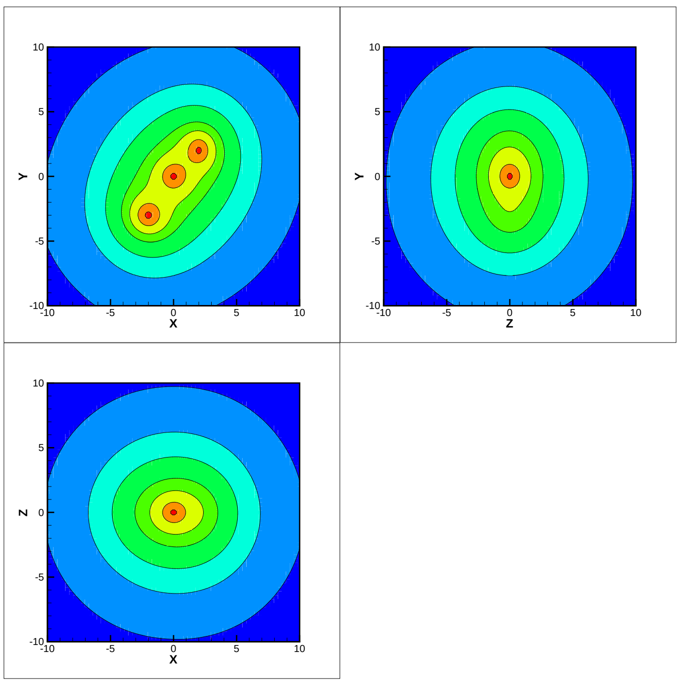

Figure 9.20: Non-singular part u of conformal factor ψ for non-time-symmetric three black holes with various mass, momentum, spin. Contour levels from outermost to innermost are $u = \{1.2, 1.3, 1.4, 1.55, 1.7, 2.0, 2.7\}$.

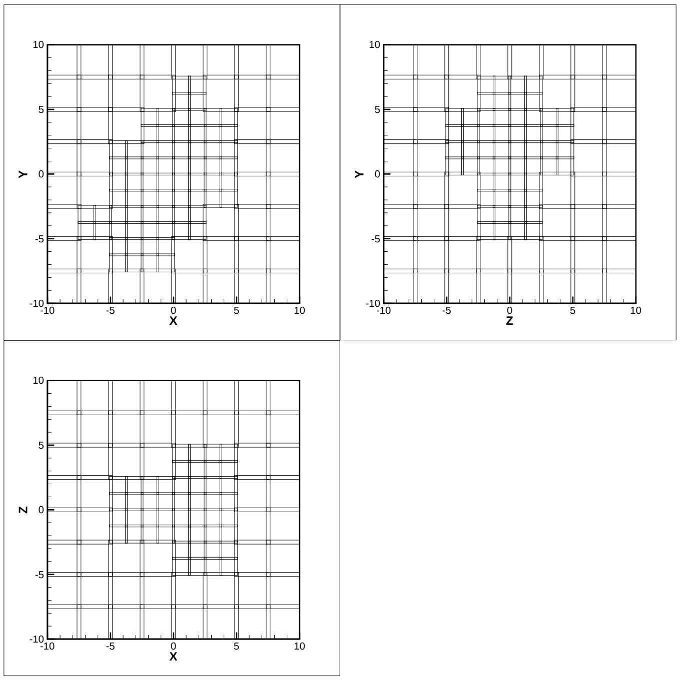

Figure 9.21: Block boundaries of adaptive mesh as found by AMRMG for non-time-symmetric three black holes with various mass, momentum, spin.

guard cell filling routine across the refinement boundaries which we use for adaptive grid spacing. The boundary guard cell filling routine is Dirichlet, and sets Φ_i at a left boundary as

$$\Phi_i = 3(\Phi_{i+1} - \Phi_{i+2}) + \Phi_{i+3} \tag{9.26}$$

and Φ_i at a right boundary as

$$\Phi_i = 3(\Phi_{i-1} - \Phi_{i-2}) + \Phi_{i-3}. \tag{9.27}$$

The Courant timestep is $c\Delta x^2$ where c is user specified. We have used $c = 0.1$ for all of our examples. When using mesh refinement, the smallest necessary timestep is calculated and used across the entire grid. The highly restrictive Courant timestep[7] leads to very slow evolutions. For example, the Misner run for $\mu = 1.8$ lasted 7.5 hours for 16,000 timesteps. This was the shortest running example shown, not including the flat space examples. Mean curvature flow with a Euclidean metric is very fast (seconds).

[7]Less restrictive timesteps were not tested.

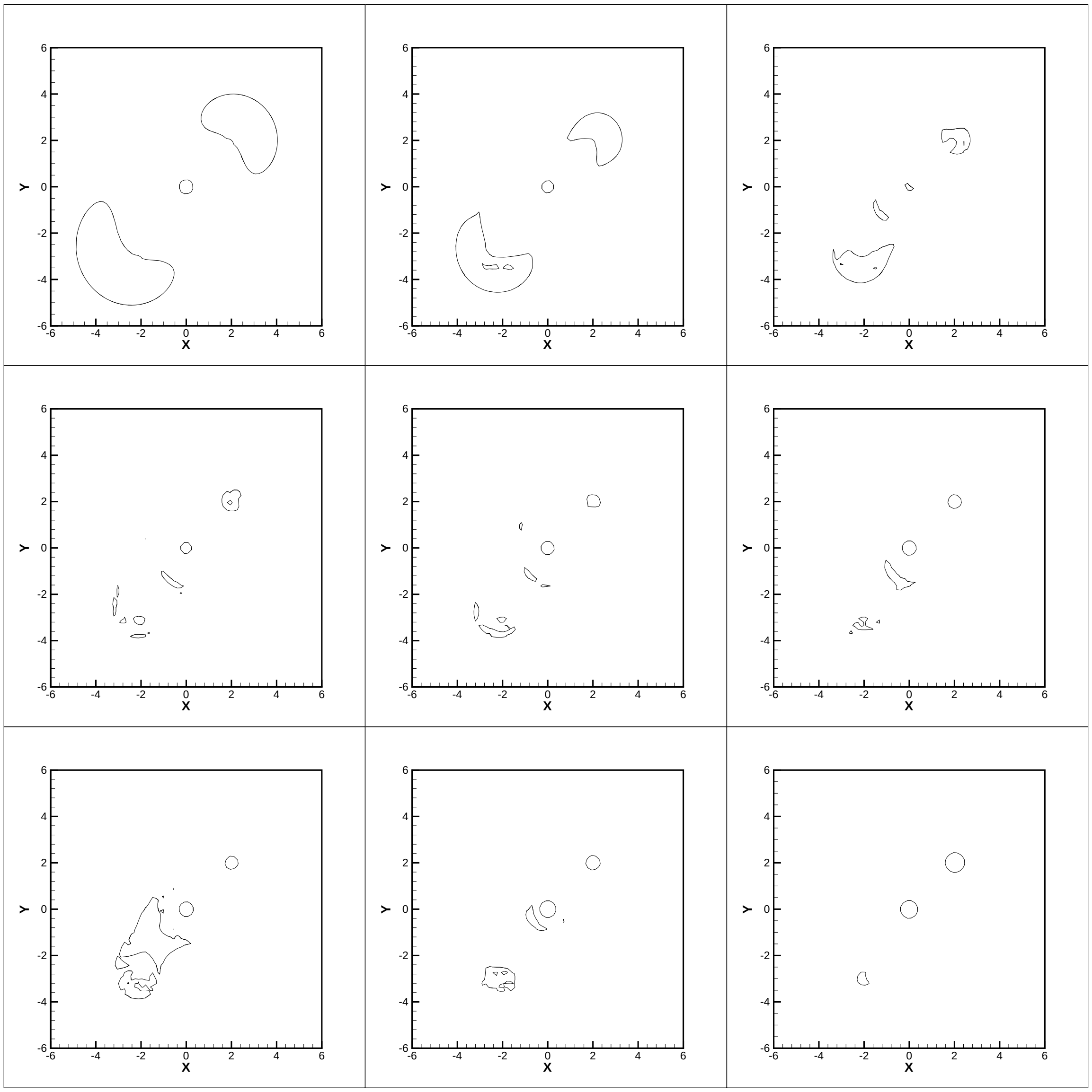

Figure 9.22: Zero level contours of $\Phi_t - \Phi_{t-1}$ for non-time-symmetric three black holes with various mass, momentum, spin.

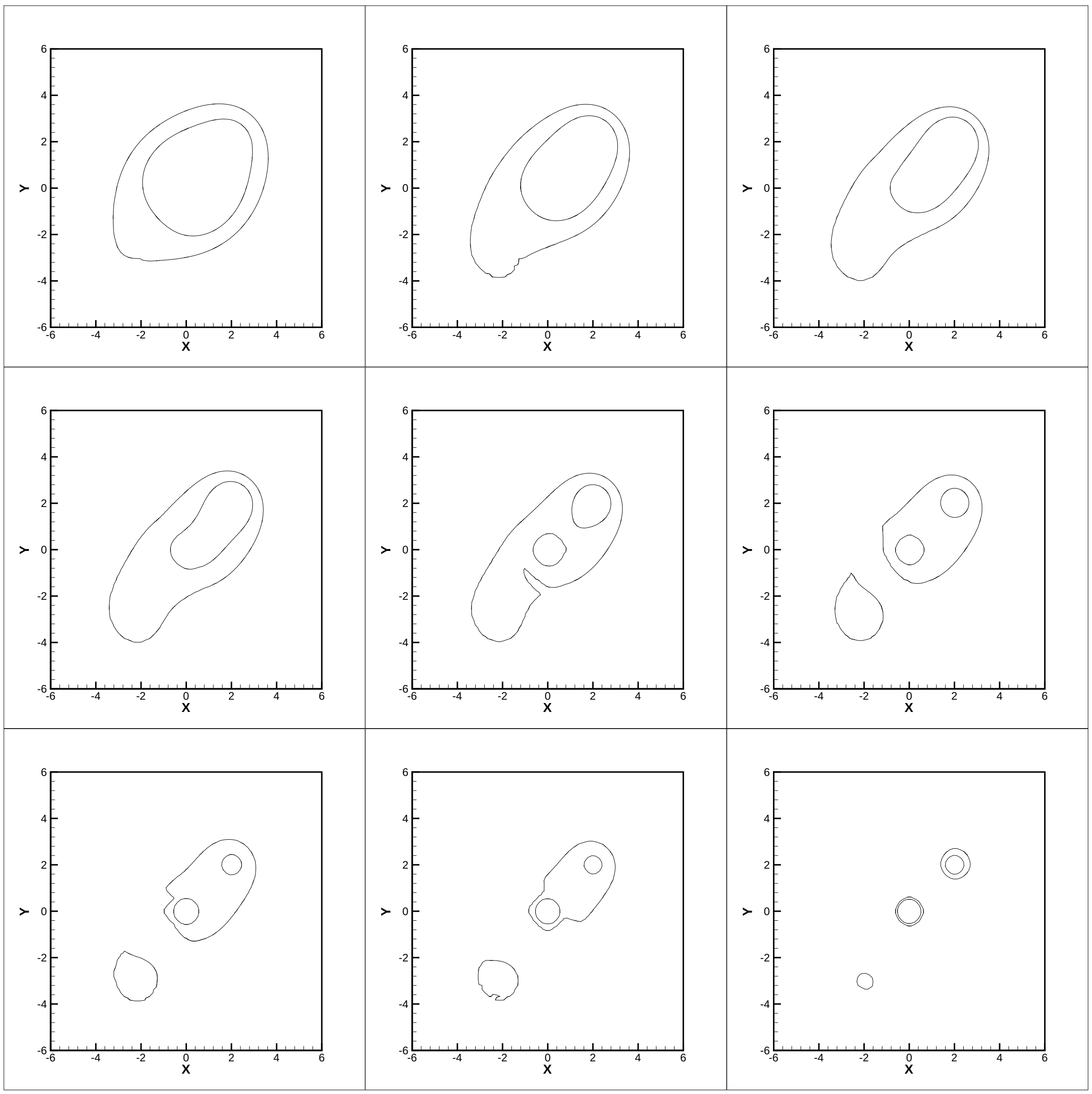

Figure 9.23: Evolution of two surfaces $\Phi = -6$ and $\Phi = -5.37$ for non-time-symmetric three black holes with various mass, momentum, spin. Last panel shows the approximate locations of the horizons.

Chapter 10

Summary and Recommendations for Future Work

In this dissertation, three numerical codes were implemented[1]. There will be a brief summary of each, with recommendations for improvements and further projects. In the following chapter, we will describe the state of the three code projects and thereby have to make use of the word 'currently'. 'Currently' will mean 'as of August, 2008'.

10.1 AMRMG—The Multigrid Solver

In chapter 4 we describe our multigrid solver using adaptive mesh refinement (AMRMG). AMRMG has been used to provide initial data for one of the first successful binary black hole coalescence and merger simulations [25]. AMRMG is a powerful computational tool for solving semi-linear elliptic partial differential equations. It has thus far only been used to provide initial data for relativity, but it could be used for many other applications. The code can be modified easily if a user wishes to solve

[1]Contact the NCSU Astrophysics Department to request access to the SVN repository containing the codes.

an equation of the form

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial y^2} + C\frac{\partial^2 u}{\partial z^2} + D\frac{\partial^2 u}{\partial x \partial y} + E\frac{\partial^2 u}{\partial x \partial z} + F\frac{\partial^2 u}{\partial y \partial z} + G\frac{\partial u}{\partial x} + H\frac{\partial u}{\partial y} + I\frac{\partial u}{\partial z} + Ju + K = \rho \qquad (10.1)$$

where all coefficients are functions of x, y, and z, e.g., $A = A(x, y, z)$.

The `GetOpsX` routines take as input `x,y,z` and give back these coefficients (above, as A–K) in the array `OPS(1:11)`. They can be used in the `GeneralRelax, General-ComputeSource, GeneralComputeResidual`, and in the `GetMatrixA` routines. If a user's equation can be cast in this form, another equation may easily be added by putting an `if(equation.eq.X)` call in these subroutines and adding another `GetOpsX` routine. For complicated functions, Maple is a useful tool for generating the coefficients in (10.1). The terms for the equations 7 and 9 (distorted black holes) were generated using a maple to Fortran command `fortran(var, optimized)` where `var` is the term that is to be converted into the fortran language.

To add an equation or problem to the multigrid code, the user must also supply boundary conditions. Robin conditions without symmetry or with bitant or octant symmetry are already included. To use a different boundary condition, one must add to the PARAMESH user routine `amr_1blk_bcset.F90` an additional `bcflag` and boundary condition definition code.

Besides adding additional equations, there are two main possible improvements to the multigrid code. The first is to rewrite the code using the new multigrid subroutines available in PARAMESH. Currently, AMRMG uses modified PARAMESH routines to carry out the multigrid operations. At the time AMRMG was written, PARAMESH was not designed with multigrid structure in mind, so the modified routines do not use memory efficiently. Replacing the current restriction and prolongation multigrid routines with the new ones written by PARAMESH should make the multigrid solver faster and more efficient. It will also make it possible to always be able to upgrade to the current version of PARAMESH without major revisions. The second possible improvement is to enable AMRMG to solve for multiple variables, such as would be necessary if the momentum constraints were to be solved in addition to the Hamiltonian constraint. This should be done making use of the PARAMESH preprocessing directives where a user specifies how many variables are to be solved. Hardwiring the code for a certain number of variables when only one may be needed will waste a considerable amount of memory.

If a future student were to make changes to improve AMRMG, the following criteria should be met before the current version of AMRMG is modified or replaced in the SVN repository:

(1) Save test cases and data files using current version of AMRMG.

(2) Run test cases using new version of AMRMG and use ‘diff’ to make sure output is *exactly* the same. If they are not the same, find the bug or confirm that they should not be the same and that the differences are expected and do not alter the performance of the code. An example of an acceptable reason for different output is a new guard cell filling routine that uses a different stencil but still maintains accuracy and high enough order convergence properties.

(3) Run the test cases with various number of processors and use ‘diff’ to make sure output is *exactly* the same for different processors. If they are not the same, find the bug or confirm that they should not be the same and that the differences are expected and do not alter the performance of the code.

(4) If the changes include porting to the new multigrid routines available with PARAMESH, check parallel scaling of the new version. If the new version does not scale well (in general, ‘scales well’ means a code runs twice as fast on twice as many processors), there may be a bug. However, even if the rewrite produces poor scaling, it may still be worth performing so as to make it easy to upgrade to new versions of PARAMESH. Confirm that it does not perform worse than our current version.

(5) Confirm that the code compiles without errors and without elaborate platform dependent caveats. Make sure a working Makefile and a runscript are included in the repository.

(6) Make a readme file of major changes, and make sure the user input routines (e.g., the file '`input`') are well documented.

(7) Check with current users of the code and with Dr. J. David Brown that the new code is still understandable and does not interfere with projects of current users.

(8) Make sure the checkpoint files are readable by the AHF code, and/or modify the AHF code in conjunction with the changes to AMRMG.

10.2 The Spectral Methods Code

A series of spectral methods routines were created in order to analyze the stability of the Einstein equations. One of the codes evolves the Einstein equations with the option of constrained evolution as described in chapter 8. Another code evolves the constraint evolution equations as described in section 6.2. The spectral methods code repository contain these codes and several others that may be useful for future graduate students for a variety of projects.

The spectral methods code repository '`SPECTRAL`' contains several directories. Each time a significant change was made, it was made to a copy of an existing code in a separate directory. The directories are as follows:

(1) `ADM`– The ADM evolution equations in 3D using spectral methods with Chebyshev polynomials.

(2) `ADMf`– The ADM evolution equations in 3D using spectral methods with Fourier polynomials.

(3) `BCG`– D. Brown's routines testing the biconjugate gradient method.

(4) `FDADMf`– The ADM evolution equations in 3D with Fourier derivative calls replaced by finite differenced derivatives.

(5) `FDLinADM`– Linearized ADM equations in 1D with finite differenced derivatives.

(6) `FiniteOneD`– Finite differencing test code solving the 1D wave equation.

(7) `LinADM`– Linearized ADM equations in 3D with Chebyshev derivatives.

(8) `LinADMOneD`– Linearized ADM equations in 1D with Chebyshev derivatives.

(9) `Spectral3D`– Spectral methods code using Chebyshev derivatives solving the 3D wave equation.

(10) `SpectralOneD`– Spectral methods code using Chebyshev derivatives solving the 1D wave equation.

(11) `TestSystems`– Integrates the constraint propagation equations using spectral methods with Fourier polynomials.

10.2.1 The Evolution Code

There are two versions of the full evolution code. The first is in the directory `ADM`. `ADM` contains the full evolution equations with derivatives given by the Chebyshev polynomials. It does not have the option for constrained evolution. The directory `ADMf` contains the full evolution equations with derivatives given by the Fourier polynomials. It has the option for constrained evolution. This is the version of the code that we have been using for the results published in chapter 8. It avoids boundary condition issues by the use of Fourier polynomials. It has automatic periodic boundaries.

10.2.2 The Constraints Evolution Code

The constraint evolution code located in directory `TestSystems` integrates the evolution equations for the Hamiltonian and momentum constraints as given by the ADM formalism. It uses Fourier polynomials. Future work may include coding and exploring the constraint evolution equations of different formalisms, such as BSSN.

10.3 AHF—The Apparent Horizon Finder

The apparent horizon finder code is currently the only implementation of a mixed mean curvature flow/level set method for solving the apparent horizon equation in non-time-symmetric spacetimes. It can find multiple black holes or horizons of any topology. AHF can find horizons given analytic forms for the metric and extrinsic curvature or from numerically generated spacetimes. It has grid infrastructure as provided by PARAMESH, and thus can be used with AMR and on multiple processors. The apparent horizon finder code needs further testing and significant improvement in speed before it can be made available to the general public. There are, however, several straightforward and relatively simple projects that can be done by future students, including undergraduates.

The first project must be done the most carefully, and that is to update to the newest version of PARAMESH. This is tricky, only because it must be done in conjunction with the multigrid solver. Before any changes with respect to PARAMESH to either AMRMG or AHF, it must be confirmed that AHF can read checkpoint files created by AMRMG. There is also currently a problem with reading in AMRMG checkpoint files using multiple processors with AHF, stemming from the last time a version of PARAMESH was changed. This problem needs to be resolved.

The second project is to experiment with various integrators to improve speed and stability. The first step in this project would be to rewrite AHF in a more object oriented fashion, such that there is a single call to a subroutine that takes the current value of ϕ and takes it through a single timestep. Once that is completed, one only has to have various `if` statements available to choose different integrators by changing an input parameter and without having to rewrite the entire code. Currently the forward backward Euler integrator as described in [79] is hard coded into the main program.

The third project is to perform rigorous testing and performance analysis. One should review the testbeds in [61] and compare results and performance. Tests should be performed to determine the long time stability properties of each integrator. In particular, how long can an integrator be used and still trust that the results are correct. Also, which integrator is the fastest, and what is the largest timestep that can be safely used. The code currently uses the same time step across the entire grid as calculated to be proportional to the smallest grid spacing. This can be altered to implement adaptive time stepping. Different initial guesses for ϕ could be reviewed and different boundary conditions explored. There is a vast repository of literature on implementing level set methods as used by the engineering community, and a literature review would be the most productive starting point in trying to improve the code.

After the above projects are complete, the code could be made user friendly and distributed. Interesting future projects could include exploring spacetime metrics that may contain toroidal (or other non-trivial topology) horizons. Projects could start with an analytic metric, or they could start by first solving the Hamiltonian constraint for a metric using AMRMG.

Appendices

Appendix A

Mean Curvature Flow– Change of Variables

Consider a two-dimensional surface S^2 in $I\!R^3$, with coordinates on the surface denoted σ^a where $a = 1, 2$. Let x^i denote a mapping from a point on the surface to a point in 3 space, or

$$x^i : S^2 \Rightarrow I\!R^3. \tag{A.1}$$

A surface may be defined as $\phi = \lambda$ where λ is a constant. Now consider a family of surfaces parameterized by λ. Each member ϕ_λ of the family of surfaces has its own set of coordinates σ^a. We will denote the family of mappings from $S^2 \Rightarrow I\!R^3$ as

$$x^i_\lambda(\sigma^a, \phi). \tag{A.2}$$

To achieve a relation between $\partial\phi/\partial\lambda$ and $\partial x^i/\partial\lambda$, consider the following mappings, assumed to be invertible:

$$\begin{aligned} x^i &= x^i_\lambda(\sigma^a, \phi) && \text{(A.3)} \\ \phi &= \phi_\lambda(x^i) && \text{(A.4)} \\ \sigma^a &= \sigma^a_\lambda(x^i) && \text{(A.5)} \end{aligned}$$

and take real number ϕ

$$\phi = \phi_\lambda\left(x^i_\lambda\left(\sigma^a, \phi\right)\right) \tag{A.6}$$

and differentiate with respect to λ to get

$$0 = \frac{\partial \phi(x^i)}{\partial \lambda} + \frac{\partial \phi_\lambda}{\partial x^i}\frac{\partial x^i}{\partial \lambda} \tag{A.7}$$

so that

$$\frac{\partial \phi_\lambda}{\partial \lambda} = -\frac{\partial \phi_\lambda}{\partial x^i}\frac{\partial x^i}{\partial \lambda}. \tag{A.8}$$

Bibliography

[1] Joan M. Centrella, Kimberly C. B. New, Lisa L. Lowe, and J. David Brown. Dynamical rotational instability at low t/w. *Astrophys. J. Lett.*, 550:193–L196, 2001.

[2] Lisa L. Lowe. Orbitmpi documentation. *PPPL 3489*, 2000.

[3] X.Z. Tang, G.Y. Fu, S.C. Jardin, L.L. Lowe, W. Park, and H.R. Strauss. Resistive magnetohydrodynamics simulation of fusion plasmas. *PPPL-3532*, 2001.

[4] J. David Brown and Lisa L. Lowe. Multigrid elliptic equation solver with adaptive mesh refinement. *J. Comput. Phys.*, 209:582–598, 2005.

[5] J. David Brown and Lisa L. Lowe. Distorted black hole initial data using the puncture method. *Phys. Rev. D.*, 70:124014, 2004.

[6] J. David Brown and Lisa L. Lowe. Modifying the einstein equations off the constraint hypersuface. *Phys. Rev. D.*, 74:104023, 2006.

[7] S. G. Hahn and R. W. Lindquist. The two-body problem in geometrodynamics. *Ann. Phys.*, 29:304–331, September 1964.

[8] Larry Smarr, Andrej Čadež, Bryce DeWitt, and Kenneth Eppley. Collision of two black holes: Theoretical framework. *Phys. Rev. D.*, 14(10):2443–2452, Nov 1976.

[9] C.M. Will. *Was Einstein Right?: Putting General Relativity to the Test*. Basic Books, New York, U.S.A., 1986.

[10] Bernard F. Schutz. *A first course in general relativity.* Cambridge University Press, 1985.

[11] R. A. Hulse and J. H. Taylor. Discovery of a pulsar in a binary system. *Astrophysical Journal*, 195:L51–L53, Jan. 1975.

[12] `http://www.ligo.caltech.edu`.

[13] Sheila Rowan Jim Hough. Gravitational wave detection by interferometry (ground and space). *Living Rev. Relativity*, 3(3), 2000.

[14] `http://lisa.jpl.nasa.gov`.

[15] `http://archive.ncsa.uiuc.edu/Cyberia/NumRel/NumRelHome.html`.

[16] Miguel Alcubierre et al. Toward standard testbeds for numerical relativity. *Class. Quant. Grav.*, 21:589, 2004.

[17] Thomas W. Baumgarte and Stuart L. Shapiro. On the numerical integration of einstein's field equations. *Phys. Rev. D.*, 59:024007, 1999.

[18] G. B. Cook et al. Boosted three-dimensional black-hole evolutions with singularity excision. *Phys. Rev. lett.*, 80:2512–2516, 1998.

[19] Steven Brandt and Bernd Brügmann. Black hole punctures as initial data for general relativity. *Phys. Rev. Lett.*, 78:3606–3609, 1997.

[20] Lee Lindblom and Mark A. Scheel. Dynamical gauge conditions for the einstein evolution equations. *Phys. Rev. D.*, 67:124005, 2003.

[21] R. Gomez et al. Stable characteristic evolution of generic 3-dimensional single-black-hole spacetimes. *Phys. Rev. Lett.*, 80:3915–3918, 1998.

[22] Miguel Alcubierre, Gabrielle Allen, Bernd Brugmann, Edward Seidel, and Wai-Mo Suen. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D.*, 62:124011, 2000.

[23] Frans Pretorius. Numerical relativity using a generalized harmonic decomposition. *Class. Quant. Grav.*, 22:425–452, 2005.

[24] Frans Pretorius. Evolution of binary black hole spacetimes. *Phys. Rev. Lett.*, 95:121101, 2005.

[25] John G. Baker, Joan Centrella, Dae-Il Choi, Michael Koppitz, and James van Meter. Gravitational wave extraction from an inspiraling configuration of merging black holes. *Phys. Rev. Lett.*, 96:111102, 2006.

[26] John G. Baker, Joan Centrella, Dae-Il Choi, Michael Koppitz, and James van Meter. Binary black hole merger dynamics and waveforms. *Phys. Rev. D.*, 73:104002, 2006.

[27] Manuela Campanelli, C. O. Lousto, P. Marronetti, and Y. Zlochower. Accurate evolutions of orbiting black-hole binaries without excision. *Phys. Rev. Lett.*, 96:111101, 2006.

[28] Joan M. Centrella. The final merger of comparable mass binary black holes. *AIP Conf. Proc.*, 873:70–81, 2006.

[29] Luis Lehner. Numerical relativity: Status and prospects. 2002.

[30] Luis Lehner. Numerical relativity: A review. *Class. Quant. Grav.*, 18:R25–R86, 2001.

[31] Miguel Alcubierre. The status of numerical relativity. 2004.

[32] Frans Pretorius. Binary Black Hole Coalescence. 2007.

[33] Mark Miller. A description of the initial value formulation of vacuum general relativity for the non-specialist.

[34] R. Arnowitt, S. Deser, and C. W. Misner. The dynamics of general relativity. 1962.

[35] Gregory B. Cook. Initial data for the two-body problem of general relativity, 1990.

[36] Y. Choquet-Bruhat and J.W. York. The cauchy problem. In A. Held, editor, *General Relativity and Gravitation: One Hundred Years After the Birth of Albert Einstein*, volume 1, pages 99–172. Plenum, New York, U.S.A., 1980.

[37] Gregory B. Cook. Initial data for numerical relativity. *Living Rev. Relativity*, 3:5, 2000.

[38] William L Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[39] P. MacNeice, K. M. Olson, C. Mobarry, R. deFainchtein, and C. Packer. A parallel adaptive mesh refinement community toolkit. *Comput. Phys. Commun.*, 126:330, 2000.

[40] `http://www.physics.drexel.edu/~olson/paramesh-doc/Users_manual/amr.html`.

[41] S. F. McCormick and J. Thomas. The fast adaptive composite grid method (fac) for elliptic equations. *Math. Comp.*, 46:439–456, 1986.

[42] Achi Brandt. Multi-level adaptive solutions to boundary value problems. *Math. Comp.*, 31:333–390, 1977.

[43] D. Bernstein, D. Hobill, E. Seidel, and L. Smarr. Initial data for the black hole plus brill wave space- time. *Phys. Rev. D.*, 50:3760–3782, 1994.

[44] Steven Brandt, Karen Camarda, Edward Seidel, and Ryoji Takahashi. Three dimensional distorted black holes. *Class. Quant. Grav.*, 20:1–20, 2003.

[45] D. Brill and R. Lindquist. Interaction energy in geometrostatics. *Phys. Rev.*, 131:471–476, 1963.

[46] Lawrence E. Kidder, Mark A. Scheel, and Saul A. Teukolsky. Extending the lifetime of 3d black hole computations with a new hyperbolic system of evolution equations. *Phys. Rev. D.*, 64:064017, 2001.

[47] Gabriel Nagy, Omar E. Ortiz, and Oscar A. Reula. Strongly hyperbolic second order einstein's evolution equations. *Phys. Rev. D.*, 70:044012, 2004.

[48] Thomas W. Baumgarte and Stuart L. Shapiro. On the numerical integration of einstein's field equations. *Phys. Rev. D.*, 59:024007, 1999.

[49] Olivier Sarbach, Gioel Calabrese, Jorge Pullin, and Manuel Tiglio. Hyperbolicity of the bssn system of einstein evolution equations. *Phys. Rev. D.*, 66:064002, 2002.

[50] Adrian P. Gentle. The BSSN formulation is a partially constrained evolution system. 2007.

[51] Alexander M. Alekseenko and Douglas N. Arnold. A new symmetric hyperbolic formulation for the Einstein equations. *Phys. Rev. D.*, 68:064013, 2003.

[52] Arlen Anderson and Jr. York, James W. Fixing Einstein's equations. *Phys. Rev. Lett.*, 82:4384–4387, 1999.

[53] John P. Boyd. *Chebyshev and Fourier Spectral Methods.* Dover, New York, 2001.

[54] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes.* Cambridge University Press, Cambridge, 1992.

[55] Lawrence E. Kidder, Mark A. Scheel, Saul A. Teukolsky, Eric D. Carlson, and Gregory B. Cook. Black hole evolution by spectral methods. *Phys. Rev. D.*, 62:084032, 2000.

[56] Harald P. Pfeiffer, Lawrence E. Kidder, Mark A. Scheel, and Saul A. Teukolsky. A multidomain spectral method for solving elliptic equations. *Comput. Phys. Commun.*, 152:253–273, 2003.

[57] Angus E. Taylor and W. Robert Mann. *Advanced Calculus.* John Wiley & Sons, Inc., New York, 1983.

[58] John Oprea. *Differential Geometry and its Applications.* Prentice Hall, Upper Saddle River, NJ 07458, 1997.

[59] Philippe Grandclement. Introduction to spectral methods. 2006.

[60] Philippe Grandclement and Jerome Novak. Spectral methods for numerical relativity. 2007.

[61] M. Alcubierre et al. Test-beds and applications for apparent horizon finders in numerical relativity. *Class. Quant. Grav.*, 17:2159–2190, 2000.

[62] Jonathan Thornburg. Event and apparent horizon finders for 3+1 numerical relativity. *Living Rev. Relativity*, 10(3), 2007.

[63] S.W. Hawking and G.F.R. Ellis. *The Large Scale Structure of Space-Time.* Cambridge Monographs on Mathematical Physics. Cambridge University Press, Cambridge, U.K., 1973.

[64] Peter Anninos, David Bernstein, Steven R. Brandt, David Hobill, Edward Seidel, and Larry Smarr. Dynamics of black hole apparent horizons. *Phys. Rev. D.*, 50(6):3801–3815, Sep 1994.

[65] J. David Brown et al. Excision without excision: the relativistic turducken. *Phys. Rev. D.*, 76:081503, 2007.

[66] M. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Diff. Geom.*, 26:287, 1987.

[67] K P Tod. Looking for marginally trapped surfaces. *Class. Quant. Grav.*, 8(5):L115–L118, 1991.

[68] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[69] L.C. Evans and J. Spruck. Motion of level sets by mean curvature, i. *J. Differential Geom.*, 33, 1991.

[70] E. Pasch. The level set method for the mean curvature flow on (r^3, g), 1997.

[71] Matthew Anderson, Luis Lehner, and Jorge Pullin. Arbitrary black-string deformations in the black string-black hole transitions. *Phys. Rev. D.*, 73:064011, 2006.

[72] Hirotaka Yoshino and Tetsuya Shiromizu. Collision of high-energy closed strings: Formation of a ring-like apparent horizon. *Phys. Rev. D.*, 76:084021, 2007.

[73] Fernando Schwartz. Existence of outermost apparent horizons with product of spheres topology. 2007.

[74] D.M. Shoemaker, M.F. Huq, and R.A. Matzner. Generic tracking of multiple apparent horizons with level flow. *Phys. Rev. D.*, 62, 2000.

[75] D.M. Shoemaker. Apparent horizons in binary black hole spacetimes, 1999.

[76] Charles W. Misner. Wormhole initial conditions. *Phys. Rev.*, 118(4):1110–1111, May 1960.

[77] D. Holz, W. Miller, M. Wakano, and J. Wheeler. Coalescence of primal gravity waves to make cosmological mass without matter. In B. Hu and T. Jacobson, editors, *Directions in General Relativity: Proceedings of the 1993 International Symposium, Maryland; Papers in honor of Dieter Brill*, pages 339–358, Cambridge, England, 1993. Cambridge University Press.

[78] Jeffrey M. Bowen and James W. York, Jr. Time asymmetric initial data for black holes and black hole collisions. *Phys. Rev. D.*, 21:2047–2056, 1980.

[79] Todd F. Dupont and Yingjie Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.*, 190:311–324, 2003.