Novel Methods to Determine and Use the Magnetic Vector Potential in Numerical General Relativistic Magnetohydrodynamics

Zachary J. Silberman
zjs2405@rit.edu

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Dissertation is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Novel Methods to Determine and Use the Magnetic Vector Potential in Numerical General Relativistic Magnetohydrodynamics

Author: Zachary J. Silberman
Advisor: Joshua Faber

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Astrophysical Sciences and Technology

School of Physics and Astronomy
College of Science

Approved by Dr. Andrew Robinson
Director, Astrophysical Sciences and Technology
The Ph.D. Dissertation of Zachary J. Silberman has been approved by the undersigned members of the dissertation committee as satisfactory for the degree of Doctor of Philosophy in Astrophysical Sciences and Technology.

________________________________________
Dr. George Thurston, Committee Chair Date

________________________________________
Dr. Joshua Faber, Dissertation Advisor Date

________________________________________
Dr. Manuela Campanelli Date

________________________________________
Dr. Matthew Duez Date
To my parents, who have always supported me in everything I do, and who introduced me to the wonders of science fiction.
ABSTRACT

Many codes have been developed to study highly relativistic, magnetized flows around and inside compact objects. Depending on the adopted formalisms, some of these codes evolve the vector potential $A$, and others evolve the magnetic field $B = \nabla \times A$ directly. Given that these codes possess unique strengths, it is sometimes desirable to start a simulation using a code that evolves $B$ and complete it using a code that evolves $A$. Transferring data from one code to another requires an inverse curl algorithm. This dissertation describes two new inverse curl techniques in the context of Cartesian numerical grids: a cell-by-cell method, which scales approximately linearly with the size of the numerical grid, and a global linear algebra approach, which lacks those ideal scaling properties but is generally more robust, e.g., in the context of a magnetic field possessing some nonzero divergence. We demonstrate that these algorithms successfully generate smooth vector potential configurations in challenging special and general relativistic contexts. In addition, we examine the magnetic helicity, which is a measure of the overall “twist” of a magnetic field configuration. It is defined as $\int_V A \cdot B \, dV$, and it should be conserved as a system evolves. By examining this quantity, we can put further constraints on the physical accuracy of numerical codes.
## CONTENTS

Abstract i  
Contents ii  
Declaration iv  
Acknowledgments v  
List of published work vi  
List of Tables vii  
List of Figures viii  

1 Introduction 1  
  1.1 Compact Objects and Their Environments 1  
    1.1.1 Observations of Gravitational Waves 3  
    1.1.2 Simulations of Relativistic Systems 4  
  1.2 Magnetism in Hydrodynamics 5  
    1.2.1 Evolving the Magnetic Field 6  
    1.2.2 Evolving the Magnetic Vector Potential 7  
    1.2.3 A Hybrid Approach 8  
  1.3 Previous Work Involving the Vector Potential 9  
    1.3.1 The Helmholtz Decomposition 9  
    1.3.2 Other Inverse Curl Techniques 11  
    1.3.3 Finite Element Analysis 12  
  1.4 Magnetic Helicity 13  
    1.4.1 Qualitative Description 13  
    1.4.2 Definition and Importance 14  
  1.5 Chapter Layout 14
I, Zachary Silberman ("the Author"), declare that no part of this dissertation is substantially the same as any that has been submitted for a degree or diploma at the Rochester Institute of Technology or any other University. I further declare that this work is my own. Those who have contributed scientific or other collaborative insights are fully credited in this dissertation, and all prior work upon which this dissertation builds is cited appropriately throughout the text. This dissertation was successfully defended in Rochester, NY, USA on July 11, 2018.

Modified portions of this dissertation have been submitted for publication by the Author in peer-reviewed paper(s) appearing in *The Journal of Computational Physics*:

- **Parts of Chapter 1, and all of Chapters 2 and 3** are based on the paper [1], entitled *Numerical generation of vector potentials from specified magnetic fields* (2018, Journal of Computational Physics, submitted), co-authored by Thomas R. Adams, Joshua A. Faber (the Dissertation advisor), Zachariah B. Etienne, and Ian Ruchlin.
ACKNOWLEDGMENTS

I would like to thank my advisor, Josh Faber, for all his help along the way, for making me a better, more independent researcher, and for working with me on a project that pushed my mental 3D visualization skills to the ultimate test. I truly believe now that I can see the Matrix . . . in our code, that is.

I also want to thank my undergraduate advisor, Jamie Lombardi, Jr., who gave me the foundation I needed to be a good researcher and programmer before I even got to graduate school, and warned me what graduate school would be like.

I would also like to thank Dennis Bowen, Manuela Campanelli, Brennan Ireland, Vassilios Mewes, Scott Noble, and Yosef Zlochower for helpful conversations regarding my work throughout my time at RIT.

And finally I want to thank all the members of the Astrophysical Sciences and Technology program, who have truly enriched this stage of my life in a wide variety of ways.
1. *Numerical generation of vector potentials from specified magnetic fields.*

LIST OF TABLES

2.1 Cell counts for each of the cell-by-cell techniques. ............................. 23
2.2 The number of equations necessary for each type of configuration. .......... 34

4.1 Values of the magnetic helicity for the rotor test. ............................. 58
4.2 Values of the magnetic helicity for the stable neutron star test. ............ 59
4.3 Values of the magnetic helicity at different grid resolutions for the stable neutron star test. ................................................................. 59
4.4 Values of the magnetic helicity for the unstable neutron star test. .......... 60
LIST OF FIGURES

2.1 Example numerical grid cell. .......................................................... 18
2.2 The sample configuration used to calculate $\alpha$ and $\beta$. ................. 25
2.3 Schematic diagram of the constraints used in matrix rows of the global linear algebra method corresponding to $A^x$ values. .................................................. 36
2.4 Schematic diagrams of the constraints used in matrix rows of the global linear algebra method corresponding to $A^y$ values. .................................................. 37
2.5 Schematic diagrams of the constraints used in matrix rows of the global linear algebra method corresponding to $A^z$ values. .................................................. 39

3.1 The components of the magnetic vector potential $A$ and the magnetic field $B$ plotted vs $x$ and $y$ at the checkpoint time $t_c$. .............................................. 44
3.2 The magnetic field $B^x$ vs $x$ at the final time $t_f$ for the rotor test. .............. 45
3.3 The magnetic field $B^z$ vs $x$ at the final time $t_f$ for the stable neutron star. .......... 47
3.4 The magnetic field $B^z$ vs $x$ at the final time $t_f$ for the unstable neutron star. .... 48
3.5 Scaling results for the cell-by-cell method. ........................................... 49
3.6 Scaling results for the global linear algebra method. .............................. 50
1.1 Compact Objects and Their Environments

Ever since Karl Schwarzschild discovered the first non-trivial solution [2, 3] to Einstein’s equations of general relativity [4], describing a non-spinning black hole (BH), astrophysicists have been interested in observing BHs, as well as delving deeper into the theory behind them and other objects that arise as solutions to Einstein’s equations.

Black holes are predicted to exist based on Einstein’s theory of gravity. They are so named because they are “black,” in the sense that they emit no light, and they are “holes” in space that matter and radiation can fall into but not get out of. More specifically, a BH is a singularity, a point with no volume that nevertheless has non-zero mass. Because of this, there is a distance within which the escape velocity from the black hole is larger than the speed of light. Near a singularity, our understanding of classical gravity breaks down, necessitating the use of general relativity to describe the system. Black holes are predicted, based on astrophysical grounds, to appear in a wide range of masses, each with their own proposed formation mechanisms. Stellar-mass BHs have masses roughly comparable to that of the Sun; that is, of order of a few $M_\odot$ or smaller, where $1 M_\odot$ is the mass of the Sun. These BHs are thought to form as the remnants of the deaths of the most massive stars. On the other hand, supermassive BHs (SMBHs) have masses $>10^6 M_\odot$, and they
reside at the centers of most galaxies. In between, we expect there to be intermediate-mass BHs (IMBHs), which bridge the gap between the previous two categories. Their formation mechanisms are still not well-understood, but they could form from smaller BHs colliding and merging together in globular clusters, which are star clusters around large galaxies that are typically composed of predominantly old stars [5].

Neutron stars (NSs), the remnants of stellar cores whose collapses were halted by neutron degeneracy pressure, are thought to form from less massive progenitors than BHs. They are sufficiently dense that the matter in them is compressed to densities at which protons and electrons recombine, leaving only neutrons and giving these stars their name [6]. Unlike BHs, NSs emit light, and can be detected by traditional observatories, often as pulsars or as a component of X-ray binaries. Pulsars are highly-spinning NSs with strong magnetic fields that emit beams of light, called jets, along their rotation axes. When such an object is oriented correctly, we receive pulses of light as the jet rotates to point toward Earth, just like a lighthouse [7]. Neutron stars can also be in binary systems, and when an “ordinary” star is in a binary system with a neutron star and is being disrupted by the gravity of the neutron star, the system is very bright in the X-ray band of the electromagnetic spectrum [8].

Such tidal disruption of stars by compact objects in binaries is an area of intense focus for astrophysicists. If an object, like a star, gets too close to a compact object, the gravitational effects from the compact object can rip apart the star, with gas unbound from the star, if not the star in its entirety, forming a gas disk around the compact object, known as an accretion disk. Accretion disks can consist of anything that got too close to the central object: stars, gas clouds, dust clouds, etc. The gas and dust in an accretion disk will be heated to high temperatures by collisional processes, potentially emitting copious amounts of high-energy radiation, such as that emitted by the X-ray binaries described above. Also, as matter from the disk falls onto the central object, some will be ejected from the system in jets, often along strongly collimated magnetic field lines wound up by dynamo-like processes. These jets, and the disks themselves, provide one avenue of detection of compact objects, regardless of whether the central object is emitting light itself. Of course, we do not want to have to rely on the presence of accretion disks to be able to detect BHs and NSs, so we
would like another way to detect them. This is a problem for which relativity supplies a solution.

1.1.1 Observations of Gravitational Waves

Another prediction of general relativity is the existence of gravitational waves, ripples in spacetime, which can be caused by, among other phenomena, mergers of compact objects such as BHs and NSs. This provides a way to detect BHs directly without relying on electromagnetic radiation, assuming we could actually detect these tiny ripples in spacetime. Einstein himself thought that we could never detect gravitational waves, and in fact 100 years passed before we were able to develop the Laser Interferometer Gravitational-Wave Observatory (LIGO) and attain a sensitivity sufficient to detect gravitational waves for the first time. LIGO is a pair of interferometers in Hanford, WA, USA and Livingston, LA, USA. Each one is composed of two arms of length four kilometers connected in a L-shape. A laser is fired and split so it goes down both arms, and mirrors at the ends bounce the light back. After the light has gone back and forth many times, the two light beams are recombined in such a way that if the two arms are the same length, the two beams cancel out. However, if the two arms are different lengths, due to, for example, a passing gravitational wave stretching and squeezing space itself, the beams will not cancel out, and we can use that signal to determine the change in the arm length. LIGO is sensitive enough to detect a relative change in length of $\sim 10^{-21}$ [9].

The first detection, denoted GW150914 (because it was detected by LIGO on September 14, 2015), was a merger between two BHs in a binary system with masses of about $36 \, M_\odot$ and $29 \, M_\odot$, putting them in the stellar-mass BH regime [10, 11]. There have been several binary black hole (BBH) detections since then, including one that was also detected by the Virgo gravitational wave interferometer in Italy [12, 13, 14, 15]. More recently, LIGO, Virgo, and $\sim 70$ observatories around the world observed a binary neutron star (BNS) merger in gravitational waves and across the electromagnetic spectrum in an event called GW170817 [16, 17].

That first BNS detection jump-started the subfield of multimessenger astrophysics. The goal of this field is to detect objects with more than one carrier of information, such as electromagnetic waves and gravitational waves, to learn as much as we can about said objects. The concept, if not
the name, of multimessenger astrophysics has been around for a while already, typically used in the past to mean electromagnetic observations paired with the search for neutrinos from astrophysical objects (see [18] for a review). The search for astrophysical neutrinos has been very successful, but there have not been any events definitively detected concurrently in neutrinos and gravitational waves (see, e.g., [19]).

While GW170817 is the first event in the history of multimessenger astrophysics, it will not be the last. We will surely observe other BNS mergers, but that is not all. Although BHs do not give off electromagnetic radiation, they could be surrounded by accretion disks giving off light. We can then try to observe binary black holes BBHs in gravitational waves and their accretion disks in electromagnetic waves to further the field of multimessenger astrophysics.

1.1.2 Simulations of Relativistic Systems

In parallel with the possibility, and then reality, of detecting gravitational waves, there has long been much interest in simulating accretion disks around BBHs so that when we detect objects we suspect are BBHs, we can compare to the simulations that have been done in an attempt to determine their physical parameters. Thus the field of numerical relativity was born, and after decades of work, multiple groups performed the first successful simulations of BBHs [20, 21, 22]. Numerical methods are required to simulate BBHs because there is no exact analytic representation of the spacetime for these systems. The techniques of numerical relativity vary according to the specific goals of the researchers. Some of the techniques are described in the next section.

Many groups are now using various numerical techniques to simulate the accretion disks around supermassive BBHs (SMBBH), which can teach us a lot about these systems and the structures of their accretion disks, such as the existence of “mini-disks” around each BH in addition to the larger circumbinary disk [23]. Such structures are much more complex than a single disk around a single object, and as we perform more and more simulations we can get a clearer picture of exactly what is going on in these systems, and what we expect to see through telescopes. Armed with these simulations, observers can work backwards from detected electromagnetic signatures to determine if the signal came from a SMBBH and if so, what its properties are. As accretion disks are made
of hot gas and dust, there could be significant effects from magnetic fields on the evolution of the system. Accurately modeling the magnetic fields is thus an important facet of these simulations.

1.2 Magnetism in Hydrodynamics

The need to consider magnetic effects in simulations of accretion disks around relativistic systems led to a new kind of computational code, building on existing formalisms and codes. The techniques of smoothed particle hydrodynamics (SPH) were already being used to simulate the evolution of single or binary stars. In SPH simulations of stars, the gas that makes up the star is split into individual chunks, called “particles,” which are then “smoothed” by describing how nearby particles interact [24, 25, 26]. In this way, one can simulate a distribution of gas as a set of discrete chunks that can be evolved individually. Because the methods of SPH do not assume anything about the configuration of the matter, they can be used to evolve accretion disks around compact objects as well. However, due to the strong magnetic fields in accretion disks, the basic SPH formalism must be adapted to include magnetic fields, leading to the development of magnetohydrodynamic (MHD) codes [27, 28]. The techniques of SPH are those of Lagrangian codes, in which fluid derivatives are calculated in the comoving frame of the fluid itself. There is a second class of codes, called Eulerian codes, in which the grid is fixed and material moves through the grid. One example of an Eulerian code is Athena [29, 30, 31], though many are used widely throughout astrophysics.

One of the major concerns in numerical evolutions of magnetic fields, for both Lagrangian and Eulerian codes alike, often as a component of a broader MHD simulation, is ensuring that the \( \nabla \cdot \mathbf{B} = 0 \) constraint of Maxwell’s equations remains satisfied. If an MHD code cannot maintain this condition, the resulting output may quickly become unphysical due to the introduction of monopoles. While directly evolving the magnetic field \( \mathbf{B} \) is an option, we must be careful to keep that field divergence-free. An equivalent technique in the case of uniform-resolution, single-patch grids involves evolving the magnetic vector potential \( \mathbf{A} \) as a fundamental variable rather than the magnetic field \( \mathbf{B} \) (see, e.g., [32]). In this case, the field can be recovered from the potential, which
is defined such that \( \mathbf{B} = \nabla \times \mathbf{A} \). The divergence of a curl is zero, so the resulting magnetic field computed from the vector potential at any point in space and time is automatically divergence-free. One issue with the vector potential arises at times when we would like to generate a vector potential configuration corresponding to a given magnetic field. Unfortunately, this can be challenging, especially if the magnetic field configuration can, in practical terms, only be represented numerically. To do so, an “inverse-curl” operator would be necessary, potentially including the specification of an electromagnetic gauge to uniquely define the resulting solution.

1.2.1 Evolving the Magnetic Field

This is not a purely academic concern; the applications of a well-defined numerical technique to generate vector potentials in specific gauges corresponding to given magnetic fields could be wide ranging, but the particular application we have in mind involves astrophysical simulation of a magnetized accretion disk about a single BH or a BBH system. The codes used to perform these simulations, such as HARM3D [33], often employ fixed or approximate background spacetime metrics, curvilinear coordinates that mimic the cylindrical nature of accretion disks, and constrained transport schemes [34, 35], in which the electromagnetic induction equations are carefully rewritten to ensure that if the divergence-free condition is satisfied initially, it is automatically satisfied at all times. In addition, these codes treat the magnetic field as a primary evolution variable. This grid structure, custom-tailored to the physical scenario, enables these codes to reliably perform extremely long-term simulations, typically of configurations that remain near quasi-equilibrium.

In the context of BBH systems, HARM3D has been extensively used to model the secular evolution of a magnetized accretion disk about the binary over timescales spanning hundreds of binary orbits (e.g. [23]). HARM3D employs an approximate spacetime metric background that is only valid when the two BHs are widely separated. As the BHs inspiral toward one another and approach merger, this approximate spacetime metric formalism breaks down, and to continue the simulation would require mapping the MHD and spacetime fields into a numerical relativity code capable of solving Einstein’s equations of general relativity without approximation, coupled to a code that solves the ideal general relativistic MHD (GRMHD) equations.
1.2.2 Evolving the Magnetic Vector Potential

For the past decade, a great deal of work has gone into constructing such an infrastructure for dynamical spacetime simulations of the mergers of BNSs or BH-NS binaries \[36, 37\], as these are likely to be important sources of gravitational waves observable by detectors such as LIGO or Virgo; see \[9, 38\] for reviews. One result of this effort has been the release of a widely-adopted open-source infrastructure of codes for general relativistic astrophysical simulations: the \texttt{Einstein Toolkit} \[32\]. Along with other Eulerian codes (see, e.g., \[39\], \[40\], \[41\]), the \texttt{Einstein Toolkit} and its constituent routines account for a substantial fraction of work in the field, with spectral codes making up much of the remainder (e.g., \[42\]). The \texttt{Einstein Toolkit} routines offer the ability to evolve the spacetime metric dynamically and use adaptive-mesh refined (AMR) Cartesian coordinates. Some groups have found that vector potential evolution schemes may be more naturally built into AMR codes than constrained transport schemes because they require fewer adaptations of the existing grid structure implementations. Obtaining values of the vector potential on a finer grid will not introduce any violation of the divergence-free condition, regardless of how those values are calculated. However, there is no such guarantee when obtaining values of the magnetic field on a finer grid directly.

Often we would prefer to evolve the vector potential $A$, as any interpolation strategy can be applied to the $A$-fields without introducing violation to the $\nabla \cdot B = 0$ constraint beyond roundoff-level. To this end, with an eye towards scenarios that arise in numerical relativity, we have in mind MHD evolutions on Cartesian grids using the \texttt{ILLINOISGRMHD} \[32\] code within the \texttt{Einstein Toolkit}. \texttt{ILLINOISGRMHD} is geared toward simulations of GRMHD fluid flows in highly-dynamical space-times, leading potentially to electromagnetic counterparts to gravitational wave signals observable by LIGO.

However, \texttt{ILLINOISGRMHD} requires that $A$ fields be specified at all grid points, and in some astrophysically-relevant contexts we are given only magnetic field data, with $A$ unspecified. Thus, we need a way to generate an $A$ field corresponding to a specified $B$ field. This is a key stumbling block, in particular for contexts in which \texttt{ILLINOISGRMHD} is the only open-source code available to continue GRMHD evolutions started by other, $B$-field based codes like \texttt{HARM3D} \[33\], which
may be used, for instance, to perform long-term MHD simulations for spacetimes that either evolve slowly or retain a high degree of symmetry.

### 1.2.3 A Hybrid Approach

One challenge in these simulations is the trade-off between physical accuracy and simulation time and resources. The more physical effects that are implemented in a numerical code, the more operations the code has to perform, causing the simulation to take longer and use more computing resources. Time and computing resources are limited, meaning researchers need to decide which effects they want to accurately model or choose to run the simulation for less time. Ideally we want to model the entire process of two compact objects merging: the initial inspiral, when the two objects are relatively far apart; the merger itself, where the two objects plunge into each other; and the ringdown, where the new, single object settles into a relaxed state. To see how to conserve computing resources while still simulating the physics accurately, we can play to the strengths of different numerical codes.

In order to maximize computational efficiency for problems involving disks around slowly-inspiraling BHs in a binary configuration, one approach would be perform the evolution with HARM3D initially, for the initial inspiral phase, while the matter configuration is closer to quasi-equilibrium, to take advantage of its curvilinear coordinates and of the approximate, analytic metric that is calculated for each timestep but not evolved by the code, cutting down on the time it takes to run a simulation. Such approximate metrics become less accurate as the BHs approach each other, so at some point we must transfer over to a full numerical GRMHD code, such as ILLINOISGRMHD, close to the merger event itself, once the spacetime metric begins evolving on a dynamical timescale. This latter code would then simulate the merger and ringdown stages of the evolution, evolving the spacetime metric as it goes. Should the former code evolve the magnetic field and the latter evolve the vector potential, the change from one code to another will require both interpolation as well as the solution of the inverse curl problem described above.
1.3 Previous Work Involving the Vector Potential

There has been interest in the inverse curl problem in the past in order to handle various phenomena. The Helmholtz decomposition (1.3.1) is a way to split a vector field into a piece that is curl-free and a piece that is divergence-free. The second term is thus of great interest when dealing with magnetic fields. There are also other methods that have been developed to calculate the inverse curl (1.3.2), and the methods of finite element analysis (1.3.3) are also useful when dealing with many operations on vector fields.

1.3.1 The Helmholtz Decomposition

The Helmholtz decomposition [43] provides a method to generate a vector potential corresponding to a given magnetic field under some assumptions which we will describe in detail below. According to Helmholtz’s theorem, any twice differentiable vector field \( \mathbf{B} \) can be decomposed into a curl-free component and a divergence-free component,

\[
\mathbf{B} = -\nabla \Psi_H + \nabla \times \mathbf{A}_H. \tag{1.1}
\]

If the field extends to spatial infinity and falls off faster than \( \mathcal{O}(1/r) \), then there are explicit expressions for both components in terms of volume integrals over all of space:

\[
\Psi_H(r) = \frac{1}{4\pi} \int \frac{\nabla' \cdot \mathbf{B}(r')}{|r - r'|} dV', \tag{1.2}
\]

\[
\mathbf{A}_H(r) = \frac{1}{4\pi} \int \frac{\nabla' \times \mathbf{B}(r')}{|r - r'|} dV'. \tag{1.3}
\]

For a finite domain \( \mathcal{V} \) with boundary \( \partial \mathcal{V} \), both integrals acquire a surface term:

\[
\Psi_H(r) = \frac{1}{4\pi} \int_{\mathcal{V}} \frac{\nabla' \cdot \mathbf{B}(r')}{|r - r'|} dV' - \frac{1}{4\pi} \oint_{\partial \mathcal{V}} \mathbf{n'} \cdot \mathbf{B}(r') \frac{dS'}{|r - r'|}, \tag{1.4}
\]

\[
\mathbf{A}_H(r) = \frac{1}{4\pi} \int_{\mathcal{V}} \frac{\nabla' \times \mathbf{B}(r')}{|r - r'|} dV' - \frac{1}{4\pi} \oint_{\partial \mathcal{V}} \mathbf{n'} \times \mathbf{B}(r') \frac{dS'}{|r - r'|}. \tag{1.5}
\]
where \( \hat{n} \) is the outward oriented normal to the surface \( \partial V \). If \( B \) is to be interpreted as the magnetic field, we know that \( \nabla \cdot B = 0 \), and the volume term in \( \Psi_H \) vanishes, leaving only the surface term. Moreover, if the magnetic field has compact support (i.e., if \( B \) vanishes on \( \partial V \)), then the integral solution for the vector potential reduces to

\[
A_H(r) = \frac{1}{4\pi} \int_V \frac{\nabla \times B(r')}{|r - r'|} \, dV', \tag{1.6}
\]

which is a solution to the vector Poisson equation

\[
\nabla^2 A_H = -\nabla \times B. \tag{1.7}
\]

There is a major disadvantage to using this method for the current problem, however. This method requires the assumption that \( B \) vanishes when approaching the boundary faster than \( O(1/r) \) in order to omit the surface integral term in Equation (1.5). If \( B \) does not fit this assumption over the numerical grid then the accuracy of the result decreases sharply near the boundary.

In cases where the magnetic field extends through the boundaries of the grid, an alternate approach is required to determine the vector potential. The term involving \( \Psi_H \) in the Helmholtz decomposition (Equation (1.4)) may be eliminated through a change of gauge. To do so, we may solve

\[
\nabla \times a_d = \nabla \Psi_H, \tag{1.8}
\]

and define a new vector potential \( A_0 \equiv A_H - a_d \). However, this requires us to perform the same inverse curl operation we already described above.

Much of the difficulty stems not so much from the problem itself, but rather from the rectangular geometry of the domain. Taking the divergence of both sides of Equation (1.8) yields

\[
0 = \nabla \cdot (\nabla \times a_d) = \nabla \cdot (\nabla \Psi_H) = \nabla^2 \Psi_H. \tag{1.9}
\]
Thus, $\Psi_H$ can be decomposed into solutions of the Laplace equation. If the domain was spherical, there would be an immediate solution available for $a_d$. We would write

$$\Psi_H = \sum_{l,m} a_{lm} r^l Y_{lm},$$

(1.10)

where $Y_{lm}$ is a spherical harmonic, and the coefficients are determined by performing an angular decomposition on the boundary. Then, we would note that

$$\nabla \Psi_H = \sum_{l,m} a_{lm} r^{l-1} \left[ l Y_{lm} \hat{r} + \Psi_{lm} \right],$$

(1.11)

where $\Psi_{lm}$ is a vector spherical harmonic. In that case, we would find that $a_d$ could be decomposed into the form

$$a_d = -\sum_{l,m} a_{lm} r^l \frac{l}{l+1} \Phi_{lm},$$

(1.12)

where $\Phi_{lm}$ is also a vector spherical harmonic.

Unfortunately, we are unaware of such a convenient set of basis functions describing solutions to the homogeneous Poisson equation for rectangular domains, and performing a spherical harmonic decomposition over a rectangular boundary is a complicated numerical task in its own right. As a result, we resort to solving for the quantity $A_0$ through both cell-by-cell and global finite differencing techniques, rather than through a spectral decomposition.

### 1.3.2 Other Inverse Curl Techniques

Several analytic and semi-analytic methods for calculating the inverse curl have been derived, but none are particularly convenient for computing the vector potential of a grid of points using Cartesian coordinates. Sahoo [44], for instance, derives fully-analytic expressions for inverse vector operations, including the inverse curl. This expression involves integrals of the components of $\mathbf{B}$ with respect to the coordinates and thus assumes a simple analytic form of $\mathbf{B}$ is known, which would greatly reduce the computational cost of this integration. In numerical work, we do not typically
have this luxury. Webb et al. [45] also find an integral expression for $A$ in terms of $B$. Their formula, in addition to requiring a smooth magnetic field, involves integrals along line segments from an origin to the point of interest, which is not computationally feasible for a simulated grid of points. In addition, the vector potential derived numerically in this method has the unfavorable quality that it is path-dependent.

Other methods of calculating the inverse curl have been found for non-Cartesian coordinate systems. Edler [46] derives expressions for the spherical harmonic coefficients of the magnetic fields. From the coefficients, the vector potential can be directly computed. However, we do not know of basis functions for Cartesian grids that are as useful for this purpose as spherical harmonics are for spherical grids. Yang et al. [47] apply a linear-algebra-based technique to extract the 12 integrals of the vector potential along the edges of a volume from the six values of $B$ on the boundaries of that volume. The values of $A$ can then be derived from these integrals using sinusoidal functions. The downside to this method is that it assumes a single volume, not small cells within a volume, so there is no way to ensure consistency from one cell to another.

### 1.3.3 Finite Element Analysis

Finite element analysis is a well-known technique for breaking a domain into small regions to facilitate the calculation of quantities such as magnetic fields. Several groups have adopted finite element analysis to solve practical problems involving magnetic fields and vector potentials, and in the process helped to develop the technique further. For example, Demerdash, Nehl, and Fouad [48] examine an air-cooled coil, and they use a tetrahedral element as their basic element. The magnetic field values are defined at the centers of the elements, and they are constant within each tetrahedron. Each element has four values for the vector potential, at the four vertices. This group divides their domain into hexahedral “super-elements,” which consist of five of these tetrahedral elements. Biro, Preis, and Richter [49] examine the differences between using nodal finite elements and edge finite elements, which is the difference between defining the vector potential at the nodes or the edges of elements. They find that using edge finite elements for the magnetic vector potential is more accurate and numerically stable than using nodal elements. In addition, they conclude
that using a specific gauge for the vector potential makes the convergence of the solution for edge elements much worse. Ren [50] shows how to construct a discrete curl matrix for a magnetic field interpolated by facet elements and a vector potential interpolated by edge elements, and then uses this matrix to solve the curl-curl equation, which relates the vector potential directly to the current density without using the field in an intermediate step.

However, these groups used finite element analysis to calculate the magnetic fields and current densities in their example setups from magnetic vector potentials. Their goals were different than those one typically has when performing grid-based Eulerian simulations, but the methods of finite element analysis are still relevant to our work. The ability to split a domain into smaller elements, of various possible shapes and sizes, is crucial to calculating the vector potential in the interior of a region, not just on its boundary.

1.4 Magnetic Helicity

1.4.1 Qualitative Description

The helicity is a quantity that describes the “twist” or “linkage” of a configuration of objects, fields, etc. To get a qualitative sense of what helicity measures, we can consider various loops of paper or ribbon, as Blackman [51] does. A Möbius strip is an example of a loop that has twist, and to increase the amount of twist one would flip the ends more times relative to each other before attaching the two ends to make a loop. However, once the ends are secured and the person making the loop lets go, the twist spreads out to the whole loop, so that there is no longer one point on the loop that is twisted. This is now a different type of helicity, called “writhe.” Another way that Blackman illustrates the difference between twist and writhe is with a roller coaster: if the track goes through a loop, that is writhe; if the car goes around the track like a corkscrew, while the track itself points in a constant direction, that is twist. Such structures in one object are also formally equivalent to two linked objects, each with half the amount of twist or writhe. This is why helicity is also a measure of the “linkage” of multiple objects. In the analogy with strips of paper, this can be shown by cutting the strip along its length. Because of the writhe in the strip,
cutting it results in two strips (of half the width) that are linked together.

### 1.4.2 Definition and Importance

In the more specific case of dealing with magnetic phenomena, magnetic helicity is defined in terms of both $\mathbf{A}$ and $\mathbf{B}$ [52]:

$$H = \int_V \mathbf{A} \cdot \mathbf{B} \, dV .$$

(1.13)

Codes like HARM3D that evolve $\mathbf{B}$ cannot calculate this quantity without first calculating $\mathbf{A}$. On the other hand, codes like ILLINOISGRMHD that evolve $\mathbf{A}$ can easily recover $\mathbf{B}$ and calculate the helicity.

Magnetic helicity is important for numerical simulations because it should be conserved for a closed system, and thus can be another check of the validity of a given simulation. In fact, when there is no dissipation, magnetic helicity is conserved even better than magnetic energy [53]. If it is going to be used in this way, it would be beneficial for any numerical code to have the capability to calculate magnetic helicity, so that its conservation can be compared between different codes. With the ability to calculate an inverse curl, even codes that evolve $\mathbf{B}$ can compute the helicity.

### 1.5 Chapter Layout

This dissertation is organized as follows: Chapter 2 outlines the two numerical algorithms we have developed to calculate the inverse curl on a staggered numerical grid, which are freely available through GitHub [54]. The first is a cell-by-cell approach that uses the values of the magnetic field to directly calculate the values of the vector potential for that cell in a symmetric way. The other technique is a global method involving large-scale sparse linear algebra solvers to solve the problem, which is in essence a linear algebra problem.

Validation, performance, and scaling tests of our codes are reviewed in Chapter 3. We have run simulations of standard GRMHD tests using ILLINOISGRMHD, stopping part-way through to substitute vector potentials output from our methods to show that our vector potentials can be
stably evolved and result in the same final magnetic fields. We also performed our methods on simulations with different numbers of grid cells to track the scaling of memory and time for our techniques.

In Chapter 4, we apply our methods to the calculation of magnetic helicity to demonstrate another use of our techniques, as well as to check the conservation of this quantity in existing numerical simulations.

Chapter 5 gives a summary of the numerical techniques we have developed and the various tests of them and their application to the calculation of helicity. We also present possible future directions for the work presented herein.
2.1 Introduction

In this chapter, we describe two techniques that can be used to perform the inverse curl operation on a staggered numerical grid, including a direct, cell-by-cell approach, as well as a global linear algebra method. The motivation for these methods comes from simplicity: how can we calculate the vector potential in a way that is inherently easy to implement numerically.

In Section 2.2 we define the structure of our numerical domain. We use a staggered grid, meaning that certain quantities are defined at some locations in the grid, while other quantities are defined at other locations. This is a standard technique, used in codes such as ILLINOISGRMHD, but it does result in a somewhat complex grid structure. However, this structure gives us simple equations by which to manipulate the fields and relate them to each other.

Then in Section 2.3, we describe the first of our two techniques. This technique is a method in which we use the values of $B$ in a given cell to generate the values of $A$ in that cell. For a single cell, this method is truly the simplest way to invert the curl. For a grid of cells, we must ensure consistency of vector potential values shared by multiple cells, which results in some complications, but these do not negate the simplicity of the method. At the end of the method, we can choose to but the vector potential into a specific gauge, and we choose the Coulomb gauge.
Chapter 2. *Code Implementation*

Following that, in Section 2.4, we detail our second technique, in which we treat the problem using linear algebra explicitly in a global way. This results in a very large matrix representing the curl operator, which also includes gauge conditions, that relates column vectors representing the A and B fields. One advantage of this method is that we can use standard, open-source linear algebra packages to solve the problem.

In Section 2.5, we address some concerns with how our techniques will interface with ILLINOIS-GRMHD. Specifically, we describe how we handled the “ghost cells,” which are extra cells placed outside the physical domain. Their purpose is to move any unphysical behavior due to the edges of the grid outside the grid itself.

We conclude the chapter in Section 2.6 with a summary of the two techniques and their respective strengths and weaknesses.

### 2.2 Vector Potentials and Magnetic Fields on Staggered Grids

#### 2.2.1 Geometry of Staggered Grids

Our grids adopt a fixed step size in each of the three Cartesian directions, and ignore complications that arise in relativity from non-constant spatial metrics. In practice, the magnetic constraint equations can always be written in terms of flat-space divergences and curls of quantities, so this assumption results in no loss of generality.

Following the standard approaches used by both constrained transport and vector potential evolution codes, we assume staggered grids for various quantities. Specifically, all hydrodynamic quantities, including fluid pressures and velocities, are known at points represented in Figure 2.1 as the centers of grid cells. We note that we reverse the standard conventions about grid-cell locations for visual clarity; one would typically describe integer-indexed quantities as the *vertices* of the grid, rather than the *centers*. Similarly, it reverses the notions of grid faces, whose values are represented in our presentation as those with two integer indices and one half-integer index, and grid edges, which here have two half-integer indices and one integer index. In order to convert back to the standard picture, in which grid cells are shifted by half a cell-width in each direction, one
must interchange centers with vertices and edges with faces. Modulo this conversion, the procedure
below remains unchanged.

Figure 2.1: Example numerical grid cell, showing the locations at which various quantities are
defined. Hydrodynamic quantities are defined at cell centers, magnetic fields at the centers of the
cell faces (for clarity, only three of the six values for this cell are labeled), and magnetic vector
potentials at the centers of cell edges. Due to the symmetry of the problem, \((i,j,k)\) may be taken
as any positively oriented cycle of the elements \((x,y,z)\) See Eqs. (2.2), (2.4), and the surrounding
text for an explanation of the index conventions for the fields.

For a numerical grid cell with index \((i,j,k)\) (where \(i\) corresponds to a unique Cartesian point
\(x\), \(j\) a unique Cartesian point \(y\), and \(k\) a unique Cartesian point \(z\)), the hydrodynamic variable
storage locations are defined as follows

\[ \rho, P, \mathbf{v} : (i,j,k) \].

(2.1)
In order to maintain divergence-free magnetic fields, we evaluate the expression $\nabla \cdot \mathbf{B}$ at grid cell centers by shifting the evaluation points for the magnetic field to cell faces,

\[
B_x^\pm : (i \pm 1/2, j, k); \quad B_y^\pm : (i, j \pm 1/2, k); \quad B_z^\pm : (i, j, k \pm 1/2).
\]

(2.2)

Then $\nabla \cdot \mathbf{B}$ at point $(i, j, k)$ is given by:

\[
\nabla \cdot \mathbf{B} = \frac{B_x^+ - B_x^-}{I} + \frac{B_y^+ - B_y^-}{J} + \frac{B_z^+ - B_z^-}{K},
\]

(2.3)

where $I$, $J$, and $K$ represent the Cartesian $x$-, $y$-, and $z$-direction grid spacings of a cell, respectively. Performing the calculation in this way, we obtain a second-order, centered finite-differencing scheme.

The divergence-free condition will be satisfied to roundoff error automatically, provided we define the vector potential $\mathbf{A}$ at the edges of each cell with staggering given by

\[
A_x^{\pm \pm} : (i, j \pm 1/2, k \pm 1/2); \quad A_y^{\pm \pm} : (i \pm 1/2, j, k \pm 1/2); \quad A_z^{\pm \pm} : (i \pm 1/2, j \pm 1/2, k),
\]

(2.4)

where we note for reasons of cyclic symmetry that one should read the $\pm$ subscripts for $A_y$ terms as representing the $z$-direction offset and then the $x$-direction offset, rather than the other way around. Then the discretized formula for the curl is

\[
B_i^\pm = \frac{A_k^{\pm \pm} - A_k^{\pm \pm}}{I} - \frac{A_j^{\pm \pm} - A_j^{\pm \pm}}{K},
\]

(2.5)

where $(i, j, k)$ represent a positively oriented cycle of the elements $(x, y, z)$—i.e., $(x, y, z)$, $(y, z, x)$, or $(z, x, y)$.

Cancellation of the numerical divergence of $\mathbf{B}$ is guaranteed, as the value of the vector potential at each edge is both added and subtracted when evaluating Equation (2.3). For example, the curl
condition on the top face of the cell yields the expression

\[
B^z = \frac{A^y_{++} - A^y_{+-} - A^x_{++} - A^x_{+-}}{I} - \frac{A^y(i + 1/2, j, k + 1/2) - A^y(i - 1/2, j, k + 1/2)}{I} - A^x(i + 1/2, j, k + 1/2) - A^x(i, j + 1/2, k + 1/2) - A^x(i, j - 1/2, k + 1/2). \tag{2.6}
\]

We note that for the purposes of bookkeeping, the quantities defined above potentially have different dimensions because of the staggering. If the grid of cells has dimension \( L \times M \times N \) in the \( x \)-, \( y \)-, and \( z \)-directions, respectively, then the numerical grid sizes of the magnetic quantities are as follows:

\[
\begin{aligned}
A^x & : L \times (M + 1) \times (N + 1), & B^z & : (L + 1) \times M \times N, \\
A^y & : (L + 1) \times M \times (N + 1), & B^y & : L \times (M + 1) \times N, \\
A^z & : (L + 1) \times (M + 1) \times N, & B^x & : L \times M \times (N + 1). \tag{2.7}
\end{aligned}
\]

### 2.2.2 Uniqueness and Gauge Choice

Determining a vector potential \( A \) that corresponds to a given magnetic field \( B \) is an under-constrained problem, and solutions are never unique. This is known as the gauge freedom of \( A \). In general, if \( \nabla \times A = B \), it is also true that

\[
\nabla \times (A + \nabla \phi) = B,
\tag{2.8}
\]

for any scalar field \( \phi \). For numerical reasons, the methods described here produce solutions in the Coulomb gauge, for which \( \nabla \cdot A = 0 \); i.e., the vector potential is also divergence-free. Numerically,
this is equivalent to the condition

\[
0 = [\nabla \cdot \mathbf{A}] (i + 1/2, j + 1/2, k + 1/2) \\
= A^x(i + 1, j + 1/2, k + 1/2) - A^x(i, j + 1/2, k + 1/2) \\
+ A^y(i + 1/2, j + 1, k + 1/2) - A^y(i + 1/2, j, k + 1/2) \\
+ A^z(i + 1/2, j + 1/2, k + 1) - A^z(i + 1/2, j + 1/2, k), \tag{2.9}
\]

where the natural location to evaluate this expression is at the grid vertices.

The Coulomb gauge is uniquely defined by requiring regularity when considering infinite domains. For finite domains, however, the Coulomb gauge solution is not unique, as one may add in gradients of the solutions of the Laplace equation that take the form \( \mathbf{a} = \nabla (r^l Y_{lm}) \), where \( r \) is the spherical radius, \( l \) is a non-negative integer, and \( Y_{lm} \) is a spherical harmonic function, without affecting either the curl or the divergence of \( \mathbf{A} \):

\[
\nabla \times \mathbf{a} = 0 \Rightarrow \nabla \times (\mathbf{A} + \mathbf{a}) = \nabla \times \mathbf{A} = \mathbf{B}, \\
\nabla \cdot \mathbf{a} = 0 \Rightarrow \nabla \cdot (\mathbf{A} + \mathbf{a}) = \nabla \cdot \mathbf{A} = 0. \tag{2.10}
\]

Given that our domains are finite, we must therefore specify additional boundary conditions for our numerical solutions. These additional conditions are discussed below.

### 2.3 Technique #1: Cell-by-Cell Generation

We present an algorithm for generating the vector potential on a cell-by-cell basis that is linearly dependent on the values of the magnetic field \( \mathbf{B} \) across each face of the cell. The divergence of \( \mathbf{B} \) is assumed to be numerically zero; if this is not the case—say because data were interpolated—we recommend use of a divergence cleaning method as a first step. Otherwise an inconsistent vector potential will be produced, as we define our staggered vector potential to produce a divergence-free magnetic field by construction. The basic outline of the cell-by-cell method follows.
To begin, choose one cell within the grid as a starting point. For convenience, we will refer to this cell as the origin, though it need not be in a corner or in the center of the grid itself. For all our tests, we started in the corner of the grid for which $i = j = k = 0$. To determine the 12 vector potential values that encircle the origin cell, we use the “six-face” technique (Section 2.3.1), which is designed to maximize the symmetry of the result with respect to the magnetic field values on the faces of the cell. We then begin scanning through the grid outward from the origin, moving cell-by-cell, row-by-row, plane-by-plane, and if we have chosen an origin that lies in the interior of the grid, octant-by-octant.

For the next cell, which we take to be one step away from the origin cell in one direction, four of the vector potential values will have been determined when we evaluated the origin cell. To determine the remaining eight vector potential values, which encircle either in whole or in part the other five faces of the cell, we use a multistep process. First, we calculate the values we would find for all 12 edges using the six-face technique, and then calculate by how much the four predetermined edges differ from these values. We then propagate the differences to the undetermined edges by modifying the six-face solution, in such a way that we reproduce the proper magnetic field on each face. As we move through the grid, we will need to apply this “five-face” technique (Section 2.3.2) on all points that lie along the coordinate axes with respect to the origin cell, permuting the ordering of the vector potential values in the function call to yield the proper rotation of our cell.

Having completed a full row along each coordinate direction, we come across a cell where the seven vector potential values encircling a pair of adjoining faces are now predetermined. We then use the “four-face” technique (Section 2.3.3) to calculate the remaining five edges, which span the other four faces in part. This method is very similar to the five-face technique. First, we calculate the six-face solution, then evaluate the differences for the seven predetermined edges, and finally propagate these changes to the remaining five edges as modifications to the six-face solution. This method will end up being applied to all cells that lie within the coordinate planes (again relative to the origin cell) outside of the coordinate axes themselves or the origin.

With the coordinate planes completed, we come across the case that describes the majority of cells within the domain, in which nine vector potential values spanning three faces have been pre-
determined, leaving only three edges that encircle in part the remaining three faces. The resulting “three-face” technique (Section 2.3.4), applied in the same manner as the previous ones, yields a global solution for the vector potential on the entire grid. In Table 2.1, we list the number of times we use each of these techniques for a grid of size $L \times M \times N$, regardless of which cell is chosen to initiate the process.

<table>
<thead>
<tr>
<th>Method</th>
<th># of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six-face</td>
<td>1</td>
</tr>
<tr>
<td>Five-face</td>
<td>$L + M + N - 3$</td>
</tr>
<tr>
<td>Four-face</td>
<td>$LM + LN + MN - 2(L + M + N) + 3$</td>
</tr>
<tr>
<td>Three-face</td>
<td>$LMN - (LM + LN + MN) + (L + M + N) - 1$</td>
</tr>
</tbody>
</table>

Table 2.1: Cell counts for each of the cell-by-cell techniques discussed below.

2.3.1 Initial Cell: Six Undetermined Faces

For the first cell in the grid that we consider, there is no prior information about the vector potential, and it must be completely determined based on the six values of the magnetic field across the faces. The problem is under-determined without making some assumptions. By assuming that whatever formulae we use to compute the vector potential will be invariant under rotations of the cube, we can obtain a unique solution.

By symmetry, there are three classes of magnetic field face values from the perspective of any particular edge: two faces of a cell immediately border the edge, two faces are located on the opposite sides of the cell from these, and the final two are the non-bordering faces at which the edge itself begins and ends. To take a particular example, for $A_{i+}^i$, the relevant magnetic field face values are $B_{j+}^j$ and $B_{k+}^k$, $B_{j-}^j$ and $B_{k-}^k$, and $B_{i-}^i$ and $B_{i+}^i$, respectively. Noting the symmetry of the problem, we need only specify one coefficient for the first pair and another for the second, as the fact that a cell has no net flux indicates that the sum of the final pair with the others equals zero.

To derive the expressions for the vector potential values, a single configuration can be constructed to establish a maximally symmetric choice for the coefficients (i.e., a choice that is not preferential towards any one direction $x$, $y$, or $z$). In Fig. 2.2, a particular magnetic field configuration is shown, in which a magnetic field of magnitude $B_{i+}^k = -4K$ can be seen as entering a
cell from the top, while the values $B^i_\pm = \pm I$, $B^j_\pm = \pm J$ represent the same field leaving the cell equally through the four sides. No flux enters or leaves through the bottom of the cell, where we set $B^k_- = 0$. For this configuration, symmetry arguments can be used to specify each of the vector potential values. Around the top face of the cell, we expect the vector potential values to all be equal (up to a common scaling factor) depending on the dimensions of the cell:

$$A^i_{++} = -A^i_{-+} = JK; \quad A^j_{++} = -A^j_{-+} = -JK,$$  

(2.11)

while those on the bottom should all be zero:

$$A^i_{+-} = A^j_{-+} = 0.$$  

(2.12)

If we assume, based on the symmetry of the problem, that the vector potential on a given edge picks up a contribution proportional to a quantity $\alpha$ for the two faces that border the edge and a contribution $\beta$ from the two on the opposite sides from these, we find, for an edge value on the top face, that

$$A^i_{++} = \alpha(KB^j_+ - JB^k_+) + \beta(KB^j_- - JB^k_-),$$

$$JK = \alpha(5JK) - \beta(JK) \rightarrow 5\alpha - \beta = 1,$$  

(2.13)

while for one on the bottom face,

$$A^i_{+-} = \alpha(-KB^j_+ + JB^k_+) + \beta(-KB^j_- + JB^k_-),$$

$$0 = \alpha(JK) + \beta(-5JK) \rightarrow \alpha - 5\beta = 0.$$  

(2.14)

The solution to this set of equations is $\alpha = 5/24$, $\beta = 1/24$, and therefore the values of the vector
Figure 2.2: The sample configuration used to calculate $\alpha$ and $\beta$. This configuration is a magnetic field that enters the top of the cell and uniformly leaves the sides. It results in the vector potential circulating around the four edges composing the top face, with the other eight vector potential values all evaluating to zero.

The potential are given by

$$
A_i^{++} = \frac{5}{24}(-K B^j_i + J B^k_i) + \frac{1}{24}(+K B^j_i - J B^k_i), \\
A_i^{-+} = \frac{5}{24}(-K B^j_i - J B^k_i) + \frac{1}{24}(+K B^j_i + J B^k_i), \\
A_i^{+-} = \frac{5}{24}(+K B^j_i - J B^k_i) + \frac{1}{24}(-K B^j_i - J B^k_i), \\
A_i^{--} = \frac{5}{24}(+K B^j_i + J B^k_i) + \frac{1}{24}(-K B^j_i + J B^k_i),
$$

where we may take any positive cyclic permutations of $x, y, z$ in terms of $i, j, k$. 

(2.15)
This solution implies that for this configuration, the vertical edges, with values $A_{\pm\pm}^k$, must be equal but could be assigned any value to yield a self-consistent magnetic field, and so we will set them to zero. This seems appropriate, as it implies that for a spatially uniform magnetic field oriented along one of the coordinate axes, the vector potential values will be purely normal with no parallel component.

### 2.3.2 Five Undetermined Faces

Once the vector potential is calculated for an initial cell, we may extend the solution to neighboring cells under the condition that the vector potential values shared with previously constructed cells are not changed.

As the solution is propagated in each coordinate direction from the initial cell, there will be cases in which one face and the four edges that surround it have already been determined, leaving five faces and eight edges yet to be determined. Call the previously determined face $B_i^-$, having denoted it $B_i^+$ when calculating vector potentials for the prior cell. For the new cell, we first compute a solution using the six-face technique described in Section 2.3.1, denoting it as $\tilde{\mathbf{A}}$. In general this will lead to conflicting results for the previously determined overlapping edges because nothing enforces consistency between $\tilde{\mathbf{A}}$ on our new cell compared to values from the previous cell. To remove inconsistencies, we propagate changes to the non-overlapping, heretofore unspecified edges. To do so, we add a constraint based on consistency. If the six-face solution $\tilde{\mathbf{A}}$ is consistent with previously computed values, we adopt it and move on. If not, we modify it on the eight undefined edges as follows.

Define $\hat{A}_{i-}^j$ and $\hat{A}_{i-}^k$ to be the vector potential values computed previously for the face encompassing $B_i^-$, and define mismatches

\[
\begin{align*}
\delta A_{\pm-}^j &\equiv \tilde{A}_{\pm-}^j - \hat{A}_{\pm-}^j; \\
\delta A_{\pm-}^k &\equiv \tilde{A}_{\pm-}^k - \hat{A}_{\pm-}^k.
\end{align*}
\] (2.16)

There are two consistent ways to determine the new edge values in a symmetric way (i.e., a way that is not preferential towards any one direction $x$, $y$, or $z$). The simplest is to propagate the
Chapter 2. Code Implementation

differences from the edges surrounding the face $B^i_-$ to those surrounding the face $B^i_+$, making sure to keep the orientation of the change correct, setting

$$A^{i+}_{\pm} = \tilde{A}^{i+}_{\pm} - \delta A^{i+}_{\pm},$$
$$A^{k+}_{\pm} = \tilde{A}^{k+}_{\pm} - \delta A^{k+}_{\pm},$$
$$A^{i\pm}_{\pm} = \tilde{A}^{i\pm}_{\pm}.$$

(2.17)

Alternately, one may change the values of the terms $A^{i\pm}_{\pm}$ on the adjoining edges to the known face, leaving the edge values of the opposing face unchanged. The symmetric form of this operation is given by the set of equations:

$$A^{i+}_+ = \tilde{A}^{i+}_+ - \frac{I}{8JK} \left[ 3(-K \delta A^{i+}_- - J \delta A^{k+}_+) + (-K \delta A^{i+}_- + J \delta A^{k+}_-) \right],$$
$$A^{i+}_- = \tilde{A}^{i+}_- - \frac{I}{8JK} \left[ 3(-K \delta A^{i+}_- + J \delta A^{k+}_+) + (-K \delta A^{i+}_- + J \delta A^{k+}_-) \right],$$
$$A^{i-}_- = \tilde{A}^{i-}_- - \frac{I}{8JK} \left[ 3(K \delta A^{i-}_- + J \delta A^{k-}_-) + (K \delta A^{i-}_- + J \delta A^{k-}_-) \right],$$
$$A^{i-}_+ = \tilde{A}^{i-}_+ - \frac{I}{8JK} \left[ 3(K \delta A^{i-}_- - J \delta A^{k-}_-) + (K \delta A^{i-}_- - J \delta A^{k-}_-) \right],$$
$$A^{j\pm}_+ = \tilde{A}^{j\pm}_+,$$
$$A^{k\pm}_+ = \tilde{A}^{k\pm}_+.$$

(2.18)

where the differences in the equations above are subject to a consistency condition:

$$J(\delta A^{k+}_- - \delta A^{k-}_-) - K(\delta A^{j+}_- - \delta A^{j-}_-) = 0.$$

(2.19)

We note that any linear combination of the two methods will also generate a consistent vector potential for the new cell as well. All results shown in this dissertation use only the first of these methods.
2.3.3 Four Undetermined Faces

Once the solution for the vector potential has been extended in each coordinate direction from the initial cell, the next step is to expand it into the coordinate planes, the planes for which one index is equal to its “origin” value. If the chosen origin is (0,0,0), these planes would be the \(xy\), \(xz\), and \(yz\)-planes. This requires solving for the vector potential for cells in which edges for two adjoining faces have been determined, leaving five leftover edges spanning parts of the four remaining faces.

Again, we begin construction of the remaining vector potential values by using the six-face method to construct a set of values denoted \(\tilde{A}\). If we assume that the faces containing the quantities \(B^i_+\) and \(B^j_-\) were previously determined, our task is to determine the new values for \(A^i_{++}\), \(A^j_{+-}\), and \(A^k_{++}\). The resulting problem is similar to the second approach for five undetermined faces, and we find

\[
A^i_{+-} = \tilde{A}^i_{+-} - \frac{3 \left( \delta A^i_{-} - \frac{I}{J} \delta A^j_{-} + \frac{J}{K} \delta A^k_{-} \right) + \left( \delta A^i_{-} - \frac{I}{J} \delta A^j_{+} + \frac{J}{K} \delta A^k_{+} \right)}{8},
\]

\[
A^j_{+-} = \tilde{A}^j_{+-} - \frac{3 \left( -\frac{I}{J} \delta A^i_{-} + \delta A^j_{-} + \frac{J}{K} \delta A^k_{-} \right) + \left( -\frac{I}{J} \delta A^i_{+} + \delta A^j_{+} + \frac{J}{K} \delta A^k_{+} \right)}{8},
\]

\[
A^i_{++} = \tilde{A}^i_{++} - \frac{3 \left( \delta A^i_{+} - \frac{I}{J} \delta A^j_{+} - \frac{J}{K} \delta A^k_{+} \right) + \left( \delta A^i_{+} - \frac{I}{J} \delta A^j_{-} - \frac{J}{K} \delta A^k_{-} \right)}{8},
\]

\[
A^j_{++} = \tilde{A}^j_{++} - \frac{3 \left( -\frac{I}{J} \delta A^i_{+} + \delta A^j_{+} - \frac{J}{K} \delta A^k_{+} \right) + \left( -\frac{I}{J} \delta A^i_{-} + \delta A^j_{-} - \frac{J}{K} \delta A^k_{-} \right)}{8},
\]

\[
A^k_{++} = \tilde{A}^k_{++}.
\]

Notice that \(A^k_{++}\) plays a slightly different role than the other edges, as it is the edge bordering a pair of faces opposite the two that have already been determined.

2.3.4 Three Undetermined Faces

The generic case describing the majority of the cells within our grid is one for which the vector potential has been determined on the nine edges encircling three adjoining faces of the cell. This leaves three undetermined edge values and three undetermined faces. If we assume that all
data have been determined on the faces supplying the values for $B_i^-, B_j^-$, and $B_k^-$, then the only remaining vector potential values left to be determined would be $A_{i+}^+, A_{j+}^+, A_{k+}^+$. Once we have determined the value of any of these three, the other two values follow by requiring $\mathbf{B} = \nabla \times \mathbf{A}$ for each face. Once one of the three values is set, there is at most one undetermined edge on any face, so they can be easily calculated.

Using the same conventions as before, we may work out the values using our initial-cell techniques, defining differences as in the previous cases, and then evaluating the symmetric formulae

$$A_{i+}^+ = \tilde{A}_{i+}^+ - \frac{1}{3} \left( \delta A_{+}^i + \delta A_{+}^i - \frac{I}{J} \delta A_{+}^j - \frac{I}{K} \delta A_{+}^k \right),$$

$$A_{j+}^+ = \tilde{A}_{j+}^+ - \frac{1}{3} \left( \delta A_{+}^j + \delta A_{+}^j - \frac{J}{K} \delta A_{+}^k - \frac{J}{I} \delta A_{+}^i \right),$$

$$A_{k+}^+ = \tilde{A}_{k+}^+ - \frac{1}{3} \left( \delta A_{+}^k + \delta A_{+}^k - \frac{K}{I} \delta A_{+}^i - \frac{K}{J} \delta A_{+}^j \right).$$

(2.21)

2.3.5 Conversion to the Coulomb Gauge and Removal of Noise

After applying this cell-by-cell algorithm, we find our solution to possess a few undesirable properties. First, nothing in these techniques will guarantee that the vector potentials we generate in a cell-by-cell way remain smooth, and indeed we find that by propagating changes across each cell in turn, we accrue rather large changes within a cell by the time we reach the edges of our grid, with potentially large jumps in the vector potential values on a cell-by-cell basis. Furthermore, the vector potential is determined in a gauge that seems arbitrary and has no convenient mathematical description. To minimize the first of these problems and eliminate the second, we convert our result to the Coulomb gauge using a convolution technique that is described below.

In general, our numerically constructed vector potential will not satisfy the Coulomb gauge condition. If we assume that we can determine a field $\phi$ to perform a gauge transformation of the form $A_C = A - \nabla \phi$, where $A_C$ is our desired Coulomb gauge solution, under the condition that
\[ \nabla \cdot \mathbf{A}_C = 0, \text{ we find} \]

\[ 0 = \nabla \cdot \mathbf{A}_C = \nabla \cdot \mathbf{A} - \nabla^2 \phi \]

\[ \nabla^2 \phi = \nabla \cdot \mathbf{A}. \quad (2.22) \]

There are numerous ways to solve the resulting scalar Poisson equation, but given the absence of well-defined boundary conditions, we choose a Fast Fourier Transform (FFT)-based convolution method. Analytically, under the assumption that \( \phi \to 0 \) as \( r \to \infty \), the solution to Equation (2.22) is given by

\[ \phi = -\frac{1}{4\pi} \int \int \int \frac{[\nabla \cdot \mathbf{A}] (\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d^3 \mathbf{r}' = \mathcal{F}^{-1} \{ \mathcal{F} \left[ \nabla \cdot \mathbf{A} \right] \ast \mathcal{F} \left[ -1/(4\pi r) \right] \}. \quad (2.23) \]

Here, the integral is evaluated over the volume of the computational domain, the symbols \( \mathcal{F} \) and \( \mathcal{F}^{-1} \) represent forward and reverse Fourier transforms respectively, and the \( \ast \) operator implies that the transformed arrays are multiplied element-by-element to perform a convolution. In practice, we do not actually convolve with the function \( -1/4\pi r \) because it will not have zero Laplacian when discretized, yielding a solution that contains a nontrivial divergence. Instead, we numerically solve the three-dimensional Laplace equation for a \( \delta \)-function source and use this as a basis for our convolution kernel.

In what follows, we will simplify our discussion by assuming that our physical grid is cubic, with \( N \) points in each dimension labeled from 1 to \( N \), with uniform grid spacing \( I \) in each of the three directions. We note, though, that these methods may be trivially generalized for cases where either the grid dimensions, grid spacings, or both are different in different directions.

In order to solve Eq. (2.23) via a convolution method, we will require a solution \( \chi \) to the equation

\[ \nabla^2 \chi = \delta(r), \quad (2.24) \]

which will serve as a convolution kernel. This kernel function must be evaluated on a grid at least
Chapter 2. Code Implementation

one cell larger in each direction than our physical grid, indexed so that the origin is placed at a
corner and the spatial domain is thought of as an octant. Thus we assume it is of size $N_{\text{ker}}^3$, where
$N_{\text{ker}} \geq N + 1$ and the indices run from 0 to $N_{\text{ker}} - 1$. Ideally, one should choose the value of $N_{\text{ker}}$
to be the smallest allowed value that can be expressed in the form $2^m + 1$, for positive integer $m$,
or a similar quantity involving small-integer factors, in order to perform the FFT described below.

The analytic solution for Eq. (2.24) is well known to be $\chi = -1/4\pi r$, where $r$ is the three-
dimensional distance from a point to the origin, $r(i, j, k) = \sqrt{i^2 + j^2 + k^2}$, but this function $\chi$
is singular at the origin and not an exact solution if we interpret the Laplacian operator as a
finite-differencing expression. Instead, we can solve

$$\chi^*(i - 1, j, k) + \chi'(i + 1, j, k) + \chi^*(i, j - 1, k) + \chi'(i, j + 1, k)$$
$$+ \chi^*(i, j, k - 1) + \chi'(i, j, k + 1) - 6\chi(i, j, k) = \begin{cases} I^2; & i = j = k = 0 \\ 0; & \text{otherwise} \end{cases} , \quad (2.25)$$

where the notation $\chi^*$ is used to describe the appropriate boundary conditions that we will apply
at the three faces of the cube describing coordinate planes, and the notation $\chi'$ is used to describe
those that we will apply at the three faces that lie on the exterior. At the former faces, we impose
reflection symmetry, such that

$$\chi^*(i, j, k) = \begin{cases} 
\chi^*(1, j, k); & i = -1 \\
\chi^*(i, 1, k); & j = -1 \\
\chi^*(i, j, 1); & k = -1 \\
\chi(i, j, k); & \text{otherwise} 
\end{cases} , \quad (2.26)$$

in order to handle cases where a neighboring value lies outside the grid across one of the coordinate
planes. At our outer boundaries, we impose a $1/r$ falloff condition:

\[
\chi'(i,j,k) = \begin{cases} 
\frac{r(N_{\text{ker}}-1,i,j,k)}{r(i,N_{\text{ker}}-1,j,k)} \chi'(N_{\text{ker}} - 1, j, k); & i = N_{\text{ker}} \\
\frac{r(i,N_{\text{ker}}-1,j,k)}{r(i,j,N_{\text{ker}}-1)} \chi'(i, N_{\text{ker}} - 1, k); & j = N_{\text{ker}} \\
\frac{r(i,j,N_{\text{ker}}-1)}{r(i,j,N_{\text{ker}})} \chi'(i, j, N_{\text{ker}} - 1); & k = N_{\text{ker}} \\
\chi(i,j,k); & \text{otherwise}
\end{cases}
\] (2.27)

though the Dirichlet condition $\chi'(i,j,k) = 0$ would also be a valid choice. The resulting linear system, which is sparse and diagonally dominant, can be solved using any standard linear algebra package. We have done so using \textsc{eigen} [55].

To perform the FFT-based convolution, we need to map the convolution kernel function and the grid containing the divergence of the vector potential into a larger grid, to avoid aliasing effects. Assuming that we have chosen the value of $N_{\text{ker}}$ appropriately, our FFT grids will have dimensions $N_{\text{fft}} \times N_{\text{fft}} \times N_{\text{fft}}$, where $N_{\text{fft}} = 2(N_{\text{ker}} - 1)$ and we assume the indices range from 0 to $N_{\text{fft}} - 1$ in each direction. The convolution kernel function is

\[
\chi_{\text{fft}}(i,j,k) = \chi(I,J,K),
\] (2.28)

where $I = \min(i, N_{\text{fft}} - i)$, $J = \min(j, N_{\text{fft}} - j)$, and $K = \min(k, N_{\text{fft}} - k)$. The data array is given by

\[
\psi_{\text{fft}}(i,j,k) = \begin{cases} 
[\nabla \cdot \mathbf{A}] (i - 1/2, j - 1/2, k - 1/2); & i < N \text{ and } j < N \text{ and } k < N \\
0; & \text{otherwise}
\end{cases}
\] (2.29)
We may now evaluate
\[
\phi_{\text{fl}} = \mathcal{F}^{-1} \{ \mathcal{F}[\psi_{\text{fl}}] \star \mathcal{F}[\chi_{\text{fl}}] \},
\]  
(2.30)
where the symbols \( \mathcal{F} \) and \( \mathcal{F}^{-1} \) represent forward and reverse Fourier transforms, and the \( \star \) operator implies that we multiply the transformed arrays against each other element by element. Our implementation uses the FFTW package [56] to perform these operations. Having calculated this field, we may then set
\[
\phi(i - \frac{1}{2}, j - \frac{1}{2}, k - \frac{1}{2}) = \phi_{\text{fl}}(i, j, k),
\]  
(2.31)
for all indices \( 1 \leq i \leq N \), \( 1 \leq j \leq N \), and \( 1 \leq k \leq N \) and then set \( A_c = A - \nabla \phi \) using our staggered coordinates, e.g.,
\[
A^x_c(i, j + \frac{1}{2}, k + \frac{1}{2}) = A^x(i, j + \frac{1}{2}, k + \frac{1}{2}) - \frac{\phi(i + \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2}) - \phi(i - \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2})}{l}.
\]  
(2.32)

The resulting vector potential configuration both satisfies the Coulomb condition in the interior and remains consistent with the given magnetic field up to machine precision levels everywhere.

### 2.4 Technique #2: Global Linear Algebra

Perhaps the most straightforward method to construct a staggered vector potential configuration corresponding to a given magnetic field is to treat the problem as a large, sparse linear algebra problem.

#### 2.4.1 Values, Constraints, and the Number of Equations

Let us first consider exactly how many equations need to be solved. According to Equation (2.7), there are \( 3LMN + 2(LM + LN + MN) + (L + M + N) \) vector potential values that must be
set. Our direct cell-by-cell method yields a count for the number of equations that must be devoted to enforcing consistency between magnetic field values on cell faces and the vector potential values spanning that face. Noting that each “n”-face method provides \( n - 1 \) consistency equations along with one that will automatically be satisfied by the divergence-free criterion, and using the cell counts shown in Table 2.1, we can get the number of equations required for each method, shown in Table 2.2. Overall, we find a total count of \( 2LMN + (LM + LN + MN) \) equations required to enforce consistency over the entire domain.

<table>
<thead>
<tr>
<th>Method</th>
<th># of Equations/Cell</th>
<th>Total Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six-face</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Five-face</td>
<td>4</td>
<td>( 4(L + M + N) - 12 )</td>
</tr>
<tr>
<td>Four-face</td>
<td>3</td>
<td>( 3(LM + LN + MN) - 6(L + M + N) + 9 )</td>
</tr>
<tr>
<td>Three-face</td>
<td>2</td>
<td>( 2LMN - 2(LM + LN + MN) + 2(L + M + N) - 2 )</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>( 2LMN + (LM + LN + MN) )</td>
</tr>
</tbody>
</table>

Table 2.2: The number of equations necessary for each type of configuration.

To construct a well-posed system, the remaining equations, numbering \( LMN + (LM + LN + MN) + L + M + N = (L + 1)(M + 1)(N + 1) - 1 \), must be specified to choose a particular gauge condition. We note that this number corresponds to the total number of grid vertices present, save one.

While any gauge condition expressible in linear form may be chosen, we will describe how to implement the Coulomb gauge condition for the sake of consistency with the cell-by-cell method. For vertices on the interior of the grid, this simply means implementing Equation (2.9), while for the vertices on the boundary we must make some assumption about the vector potential components that lie outside of the computational domain. We choose a simple, nonsingular\(^1\) option: zero the normal components of the vector potential at the boundary of the domain.

While it is straightforward to construct a sparse matrix problem whose solution is the desired vector potential configuration, computational efficiency concerns force us to consider the organization of the linear system. Unlike the case for calculating the convolution kernel for Coulomb gauge

\(^1\)We note that a “copy”-type boundary condition, in which the normal components of the vector potential are set to be equal to those immediately inside the boundary, yields a singular linear system that cannot be evaluated, as it permits a constant non-zero vector potential solution for zero source.
conversion described in Section 2.3.5, this matrix cannot be made diagonally dominant, nor can it
easily be constructed in symmetric form, so several efficient techniques for solving sparse systems
are immediately ruled out. We have found that most solvers perform better when the diagonal
terms are non-zero in each row, and when the bandwidth of the non-zero elements is minimized,
which motivates our approach to the problem. In what follows, we discuss a straightforward ap-
proach to generating a sparse linear system that yields a vector potential solution in the Coulomb
gauge, satisfying the principles above.

2.4.2 Constructing the Linear System

Our linear system consists of one equation for each of the unknown vector potential values:
a total of \(3LMN + 2(LM + LN + MN) + L + M + N\) equations. Each of the vector potential
components is organized in dimension-by-dimension order, with the \(z\)-coordinate varying most
rapidly and the \(x\)-coordinate the slowest. In order to maximize geometric proximity of neighboring
elements in our linear system, we recommend interleaving the vector potential components; thus,
successive rows of our matrix represent \(A^x\), \(A^y\), and \(A^z\) components in turn (this is easiest for cubic
grids, for which the different vector potential components contain the same number of elements on
the grid).

2.4.2.1 Equations for \(A^x\)

The first set of linear equations is found by associating each \(A^x\) value with a corresponding
instance of the Coulomb gauge condition. In particular, the matrix row corresponding to a value
\(A^x(i, j + \frac{1}{2}, k + \frac{1}{2})\) is associated with the equation

\[
0 = [\nabla \cdot A](i + \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2}) ,
\]

(2.33)

where the right-hand side is evaluated using Equation (2.9). A schematic diagram of this process
is shown in Figure 2.3. Again, any vector potential value lying outside the computational domain
is set to zero. This has the effect of enforcing the Coulomb condition at every grid vertex in
the domain except those on the “leftmost” face with coordinates \((-1/2, j + 1/2, k + 1/2)\), which are handled later.

\[
A_x^r[0,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[1/2,-1/2,-1/2] \Leftrightarrow A_x^r[1,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[3/2,-1/2,-1/2] \Leftrightarrow A_x^r[1,1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[3/2,1/2,-1/2]
\]

\[
A_y^r[0,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[1/2,-1/2,-1/2] \Leftrightarrow A_y^r[1,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[3/2,-1/2,-1/2]
\]

\[
A_z^r[0,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[1/2,-1/2,-1/2] \Leftrightarrow A_z^r[1,-1/2,-1/2] \Leftrightarrow (\nabla \cdot A)[3/2,-1/2,-1/2]
\]

Figure 2.3: Schematic diagram of the constraints used in matrix rows of the global linear algebra method corresponding to \(A_x^r\) values: Coulomb conditions for all values. See text for full description of these calculations.

2.4.2.2 Equations for \(A_y^r\)

For the rows corresponding to \(A_y^r\) values, we use two different types of equations. For the leftmost set of values, i.e., those with \(i = -1/2\), the Coulomb condition is enforced, such that for the row corresponding to \(A_y^r(-1/2, j, k + 1/2)\), our matrix row implements the equation

\[
0 = [\nabla \cdot A](-1/2, j + 1/2, k + 1/2),
\]

(2.34)

with the same treatment as above for components lying outside the computational domain. This is shown in Figure 2.4a. Combined with the previous step, the only vertices at which the Coulomb condition has not been applied are those on the grid edges satisfying the condition \(i = j = -1/2\).

For the remaining rows corresponding to \(A_y^r\) values, we demand consistency with the \(B_z^r\) value for the given magnetic field. In particular, for the row corresponding to \(A_y^r(i + 1/2, j, k + 1/2)\), we implement the equation

\[
B_z^r(i,j,k+1/2) = \frac{A_y^r(i+1/2,j,k+1/2) - A_y^r(i-1/2,j,k+1/2)}{I} - \frac{A_x^r(i,j+1/2,k+1/2) - A_x^r(i,j-1/2,k+1/2)}{J},
\]

(2.35)

See Figure 2.4b for a schematic diagram of this step.
Figure 2.4: Schematic diagrams of the constraints used in matrix rows of the global linear algebra method corresponding to \( A^y \) values: (a) Coulomb conditions on the left-most face \((i = -\frac{1}{2})\), and (b) Consistency with \( B^z \) values elsewhere. See text for full description of these calculations.

### 2.4.2.3 Equations for \( A^z \)

For rows corresponding to \( A^z \) values, three sets of equations must be implemented:

1. For those on the edge of the domain, with coordinates \( i = j = -\frac{1}{2} \), the Coulomb gauge condition is applied, so that for the row corresponding to \( A^z(\frac{-1}{2}, -\frac{1}{2}, k) \), we implement

\[
0 = (\nabla \cdot A)(\frac{-1}{2}, -\frac{1}{2}, k + \frac{1}{2}) .
\]  

(2.36)

The schematic diagram of this step is shown in Figure 2.5a. At this point, the Coulomb condition has been enforced at every vertex within the domain, except the corner point with coordinates \( i = j = k = -\frac{1}{2} \). This serves as the single point at which we do not have the degrees of freedom available to implement the Coulomb condition.

2. For remaining \( A^z \) values with coordinates \( i = -\frac{1}{2} \), we enforce consistency for given values of
In particular, for rows corresponding to the values $A^z(-1/2, j + 1/2, k)$, we implement

$$B^x(-1/2, j, k) = \frac{A^z(-1/2, j + 1/2, k) - A^z(-1/2, j - 1/2, k)}{J} - \frac{A^y(-1/2, j, k + 1/2) - A^y(-1/2, j, k - 1/2)}{K}.$$  \hfill (2.37)

See Figure 2.5b for a schematic diagram corresponding to this step.

3. For the remaining rows corresponding to $A^z$ values lying elsewhere, with $i \neq -1/2$, we enforce consistency for the given $B^y$ values. Rows corresponding to values $A^z(i + 1/2, j + 1/2, k)$ are used to solve the equations

$$B^y(i, j + 1/2, k) = \frac{A^x(i, j + 1/2, k + 1/2) - A^x(i, j + 1/2, k - 1/2)}{K} - \frac{A^z(i + 1/2, j + 1/2, k) - A^z(i - 1/2, j + 1/2, k)}{I}.$$  \hfill (2.38)

Figure 2.5c contains the schematic diagram for this last step.

### 2.4.3 Solving the System

The linear system of equations was implemented in the MUMPS [57, 58, 59] infrastructure, an MPI-parallelized sparse matrix package. With it, we are capable of generating suitable vector potentials even on numerical grids of order $100^3$ in size, as discussed and demonstrated in Chapter 3.

For input magnetic field configurations that are divergence-free, our method yields the unique vector potential consistent with both it as well as the gauge and boundary conditions. For a magnetic field that does contain numerical divergences, our method acts as an implicit one-dimensional “divergence cleaner”, sweeping numerical divergences off of the grid in a cell-by-cell fashion, working from the $B^x$ values on the leftmost face of the grid and transferring away divergences to the right and eventually out of the rightmost face of the grid. Our choice of the roles of the $x$-, $y$-, and $z$-directions is arbitrary, chosen simply for convenience.
Chapter 2. *Code Implementation*

Figure 2.5: Schematic diagrams of the constraints used in matrix rows of the global linear algebra method corresponding to $A^z$ values: (a) Coulomb conditions on the left-most edge ($i = j = -1/2$), (b) Consistency with $B^x$ values in places where $i = -1/2$ and $j \neq -1/2$, and (c) Consistency with $B^y$ values elsewhere. See text for full description of these calculations.
2.5 Interface with IllinoisGRMHD

ILLINOISGRMHD explicitly creates ghost cells outside the grid. However, the cell-by-cell and global linear algebra methods do not take this into account. They treat all the data as if they are part of the physical grid. Without any adjustment, this causes inconsistencies when data are fed back into ILLINOISGRMHD because by default, ILLINOISGRMHD linearly extrapolates data in the ghost cells. This is an issue outside of the B to A calculation; it is purely related to the interface with ILLINOISGRMHD. Therefore it is valid to handle the ghost cells independent of the B to A calculation.

To achieve matching behavior at the boundaries between our methods and ILLINOISGRMHD, it is necessary to treat different ghost zones in different ways. This is due to staggering of the data in our methods and ILLINOISGRMHD, as described in Section 2.2. If we just extrapolated all A-values linearly to match the ILLINOISGRMHD default, we would introduce errors into the condition that \( \mathbf{B} = \nabla \times \mathbf{A} \). Therefore we use hybrid quadratic-cubic conditions. In this scheme, ghost values are handled as follows: normal A-field components are extrapolated quadratically from data within the physical grid, and tangential A-field components are extrapolated cubically.

For example, if we are looking at the ghost cells on the +x-side of the grid, then we would have

\[
A^x(i, j, k) = 3A^x(i - 1, j, k) - 3A^x(i - 2, j, k) + A^x(i - 3, j, k)
\]

\[
A^y(i, j, k) = 4A^y(i - 1, j, k) - 6A^y(i - 2, j, k) + 4A^y(i - 3, j, k) - A^y(i - 4, j, k)
\]

\[
A^z(i, j, k) = 4A^z(i - 1, j, k) - 6A^z(i - 2, j, k) + 4A^z(i - 3, j, k) - A^z(i - 4, j, k),
\]

(2.39)

where, if the number of ghost cells on either side of the grid is \( n_{\text{ghost}} \), i runs from \( L + n_{\text{ghost}} - 1 \) to \( L + 2(n_{\text{ghost}} - 1) \), inclusive, and j and k cover their full range of values, including ghost cells.

2.6 Summary

In this chapter we described the two methods we have developed to calculate the vector potential A from the magnetic field B, or more generally, the inverse curl of any vector field. The first is a
cell-by-cell method in which the values of the vector potential in a cell are linearly dependent on the values of the magnetic field in that cell. The other is a method that treats this problem like the linear algebra problem it is.

The cell-by-cell method is in some respects the simplest way to invert a curl, especially for a single cell. A full grid introduces the need for consistency considerations, but that does not detract from the ease of implementation. The calculation of the values of $\mathbf{A}$ directly from linear combinations of the values of $\mathbf{B}$ is the essence of this technique, and it results in very simple equations that must be implemented. This method does require a divergence-free magnetic field as input, which should not be a problem, except for the possibility of non-zero divergence due to interpolation. This method also results in a vector potential that is in an arbitrary gauge, so if a specific gauge is desired, a gauge transformation must be performed afterwards.

On the other hand, the global linear algebra method is slightly harder to implement but has better symmetry properties and includes the gauge conditions inherently. Because this is a global method, as opposed to working through the grid one cell at a time as in the first method, the resulting vector potential is more symmetric and does not depend on the order in which the values of the vector potential are calculated. In addition, the matrix we construct takes care of any non-zero divergence in $\mathbf{B}$ by sweeping that divergence through the grid and eventually out one of the sides of the grid. The Coulomb gauge conditions are also included in the matrix, eliminating the need to do a separate transformation.

In the next chapter, we present various tests of our methods, which show the validity of our methods, as well as the performance and scaling of each method. These tests give us other ways to compare and contrast the two techniques.
3.1 Introduction

We have run a variety of tests on our codes to ensure that they can be used in situations like those that provided the motivation for their development. First we perform tests that check to see if the vector potentials we generate are consistent with the given magnetic fields and if our vector potentials can be used in GRMHD simulations such that the vector potentials still result in consistent magnetic fields after being evolved. After these validation tests, we also performed performance tests to compare how long the different techniques take to complete and how much memory each requires.

To test the validity of our codes, we use the ILLINOISGRMHD\(^1\) code within the EINSTEIN TOOLKIT as our dynamical evolution code for various magnetized fluid configurations, as it evolves the magnetic vector potential over time as its dynamical variable. For each configuration, we perform the simulation twice.

First, we run ILLINOISGRMHD to the designated final time as a baseline. We call this the “uninterrupted run”. Next we start a second, “restarted” run, in which the simulation proceeds to

\(^1\)See Section 2.5 for a discussion of how ghost cells are treated in the context of interfacing our methods with ILLINOISGRMHD.
a predetermined “checkpoint” time. At this point, we output the A field and numerically compute a B field from it (via the definition of A) and use these magnetic fields as input for our cell-by-cell and global-linear-algebra codes. We then generate a new vector potential A′ from the B field, which will typically differ significantly from those generated by ILLINOISGRMHD. We feed A′ back into ILLINOISGRMHD and restart the run at the checkpoint time and continue until the final time.

We expect that if our approach is valid, the final magnetic fields (i.e., the physical, as opposed to the gauge-dependent, fields) will agree to many significant digits between the restarted or uninterrupted runs. What follows is a demonstration of the validity of our approach in a variety of contexts: a magnetized two-dimensional, relativistically-spinning rotor (Section 3.2), a weakly-magnetized NS (Section 3.3.1), and an extremely-magnetized NS (Section 3.3.2).

Then in Section 3.4, we present the results of timing how long it takes the different techniques to calculate the inverse curl for different numerical resolutions of the weakly-magnetized NS. This section also presents the scaling relation of each technique in regard to the amount of memory used for different resolutions.

The chapter concludes in Section 3.5, summarizing the tests performed and comparing the two techniques in the context of the test results.

### 3.2 Rotor Test

The rotor test, as described by Mösta et al. [60], consists of a disk of material in the xy-plane, referred to as a rotor, which possesses ten times the density of the surrounding material. The rotor is spinning such that its edge is moving at 0.995c, where c is the speed of light, and the entire medium is permeated by a uniform magnetic field in the x-direction. We run the simulations to a final time (t_f) at which the rotor has undergone two-thirds of a full rotation, with a checkpoint time (t_c) of one-third a full rotation.

Figure 3.1 shows the components of the magnetic vector potential A and those of the magnetic field B in arbitrary units, plotted in the xy-plane at the checkpoint time. These are data from the output of the cell-by-cell method. By inspection, these fields appear fairly smooth in their own
right, but are they consistent with those of the uninterrupted run?

Figure 3.1: The components of the magnetic vector potential $\mathbf{A}$ (Left) and the magnetic field $\mathbf{B}$ (Right) plotted vs $x$ and $y$ at the checkpoint time $t_c$ after running through the cell-by-cell method. All quantities are in arbitrary units. Note that $B^z$ is zero everywhere.

To compare the two runs of ILLINOISGRMHD, we look at the components of the magnetic field $\mathbf{B}$. We do not directly compare the vector potential $\mathbf{A}$ between the two runs because the
chosen vector potential gauges are different, depending both on whether an inverse curl algorithm is applied and on which such algorithm is chosen. Thus the final $A$ field can be very different between uninterrupted and restarted runs, yet still yield the same magnetic field. This is why we say the magnetic field is the physically relevant quantity. Figure 3.2 displays the dominant magnetic-field component, $B^x$, versus $x$ at the final time for all three runs: uninterrupted, restarted with cell-by-cell data, and restarted with global linear algebra data. Notice the agreement is within about one part in $10^8$ or more throughout the entire data set.

![Magnetic Field](image)

Figure 3.2: Top: The magnetic field $B^x$ vs $x$ at the final time $t_f$ after running the rotor through the cell-by-cell and global linear algebra methods. Bottom: The absolute difference between the cell-by-cell and global linear algebra runs and the corresponding uninterrupted run.
3.3 Magnetized Neutron Stars

Here we consider models of NSs with interior, poloidal magnetic fields, described using the Tolman-Oppenheimer-Volkhoff (TOV) model, a configuration that is also described by Mösta et al. [60]. In Section 3.3.1, we start with a magnetic field strength parameter of $A_b \approx 0.64$, which results in a ratio of magnetic pressure to gas pressure of $b^2/2P \sim 0.001$. This should result in a star with little to no evolution due to magnetic effects, so we call this the “stable” case. Then, in Section 3.3.2, we set the initial magnetic field strength a value $10^2$ times larger, resulting in a magnetic to gas pressure ratio $10^4$ times larger. This should cause the magnetic pressure to violently blow apart the star, so we refer to this as the “unstable” case. The timescale of interest in these tests is the dynamical timescale of the stable NS, $t_{\text{dyn}}$. The final time for all tests is fixed at $4t_{\text{dyn}}$, and the checkpoint time is set to $t_{\text{dyn}}$.

3.3.1 Magnetically Stable Case

As expected, there was not much evolution of the magnetic, hydrodynamic, or gravitational fields in the “stable” configuration. The NS slowly relaxes from its initial state. Figure 3.3 shows the dominant magnetic field component, $B^x$, along the $x$ axis for the cell-by-cell and global linear algebra runs at the final time $4t_{\text{dyn}}$. Conventions are the same as in Figure 3.2. Agreement between runs is exceptional, particularly in the region of interest: the interior of the star ($|x| \lesssim 0.65$) where the magnetic fields remain nonzero throughout the evolution.

However, small disagreements exist at the boundary, due to the hybrid quadratic-cubic boundary conditions described in Section 2.5. These boundary conditions pose some difficulty for numerical evolutions, as even the uninterrupted case shows the development of magnetic phenomena near the grid boundary. Also note this effect is only present on one side of the grid, when $x \approx -2$. We believe this to be due to ILLINOISGRMHD’s extrapolation of $A$ to fill in ghost cells at the boundary. In a full-scale simulation, the boundary will be sufficiently far away that such edge effects will not be a problem.
Figure 3.3: Top: The resultant $B_z$ vs $x$ after running the stable NS through the cell-by-cell (CBC) and global linear algebra (GLA) routines and restarting. Bottom: The absolute difference between the cell-by-cell and global linear algebra runs and the corresponding uninterrupted run. The checkpoint time was the dynamical time, and the final time was four times the dynamical time.

3.3.2 Magnetically Unstable Case

Our inverse curl algorithms also performed well in this test, though agreement dropped to about one part in $10^4$, likely due to the transmission of errors from the boundary (the simulation lasts $\approx 9$ light-crossing times). Over the course of the simulation, the star blew itself apart, ejecting material towards the edge of the grid. The data for $B_z$ vs $x$ are displayed in Figure 3.4 using the same conventions as Figure 3.3 in the previous section. Here the data on both ends of the grid ($x \approx -2$ and $x \approx 2$) actually match each other better than in the stable case. We think this is due to the behavior seen in the stable case being wiped out by the error caused by the outflow of the
star exploding.

![Magnetic Field Graph]

Figure 3.4: *Top:* The resultant $B^z$ vs $x$ after running the unstable NS through the cell-by-cell and global linear algebra routines and restarting. *Bottom:* The absolute difference between the cell-by-cell (CBC) and global linear algebra (GLA) runs and the corresponding uninterrupted run. The checkpoint time was the stable dynamical time, and the final time was four times the stable dynamical time.

### 3.4 Code Performance

We have tested the performance of our codes on the NewHorizons computational cluster at RIT. Overall, it consists of 64-bit AMD and Intel CPUs interconnected with a high-speed, low-latency QDR InfiniBand fabric, and 4 GB of RAM per core. Specifically, our simulations were run on a subset of NewHorizons with dual-core AMD Opteron™ processors.
Chapter 3. Numerical Tests

Our cell-by-cell method is written as a serial code. As shown in Figure 3.5, the computational time it requires scales linearly with the number of grid cells, and can be expected to follow this behavior even for much larger grids. Because the cell-by-cell method is strictly serial, it must be run on a single core, so the CPU core time is equal to the walltime. The code converges for all resolutions on a single core. It requires very little memory to generate its initial vector potential configuration beyond that required to store the magnetic field and vector potential values. Larger grids are required to perform the FFT convolution while converting the vector potential to the Coulomb gauge, as discussed in Section 2.3.5, but if need be this step could be separated from the rest of the code and run in parallel using the existing FFTW infrastructure [56], which is the leading MPI-parallelized multi-dimensional FFT package.

![Stable NS/Cell-by-Cell Method](image)

Figure 3.5: Scaling results for the cell-by-cell method, as measured in walltime versus number of grid cells, for the stable NS test.
Figure 3.6 shows that the walltime required to use our global linear algebra method largely scales as the number of gridpoints, $N$, to the $5/3$ power, as expected due to how MUMPS implements basic Block-Low Rank (BLR) factorization [61]. The only significant deviation from this pattern appears at large grid sizes, for cases in which RAM runs out and swapping occurs. More minor deviations from this pattern occur due to matrix inversion requiring a significant communication overhead (handled internally by the MUMPS package). Thus the use of additional cores typically increases the required CPU core time to complete a run, even if walltime is reduced. In terms of memory, this method scales as $N$ to the $4/3$ power.

![Stable NS/Global Linear Algebra Method](image)

Figure 3.6: Walltime and CPU core time for the global linear algebra method run on different numbers of cores on NewHorizons for the stable NS test. For a given number of CPU cores, the solid line represents the walltime, and the dashed line represents the CPU core time. Points marked with $\times$’s correspond to the highest resolution $N$ for which the code converges with the given amount of RAM (4GB/core on NewHorizons).
3.5 Summary

In this chapter we presented tests of the two solution methods presented in Chapter 2, using standard tests, including the rotor test and evolution of magnetized NSs. For both tests, we find that if we evolve the system in ILLINOISGRMHD using \textbf{A} fields calculated via either of our techniques, the resulting \textbf{B} fields match the results of evolving the systems with the native \textbf{A} fields to a high degree of accuracy. In many cases, it is hard to say which of our techniques results in fields that better match those of the original simulation.

While the validation tests do not indicate any large advantages of using one of our techniques over the other, the performance tests clearly indicate that each method has its own strengths and weaknesses. The cell-by-cell approach scales linearly, in both time and memory, with the number of grid cells \( N \). On the other hand, the global linear algebra method scales as \( N \) to the \( 5/3 \) power in time and \( N \) to the \( 4/3 \) power in memory. As a counterpoint to these scaling relations, it should be noted that the cell-by-cell method is a strictly serial code, while the global linear algebra method is fully parallelized. Nevertheless, the serial cell-by-cell technique takes overall less time than the parallelized global linear algebra method.
CHAPTER 4

APPLICATION TO MAGNETIC HELICITY

4.1 Introduction

As stated in Chapter 1, helicity describes the twist and writhe of a single object or the linkage of multiple objects. The magnetic helicity can be used to help understand the characteristics of a magnetic field configuration, and should be conserved extremely well in MHD simulations. In this chapter we show how to calculate the magnetic helicity in the context of GRMHD simulations, making use of our inverse curl methods described in Chapter 2.

In Section 4.2, we go through the basics of how to calculate the magnetic helicity, working off its definition (Equation 1.13), which is given again here:

\[ H = \int_V \mathbf{A} \cdot \mathbf{B} \, dV. \]  

(4.1)

In Section 4.3, we give the results of our calculations of the magnetic helicity at different times during GRMHD simulations to determine the level of helicity conservation. The chapter ends in Section 4.4 with a summary of the calculation, use, and interpretation of magnetic helicity.
4.2 Calculation of Helicity

In order to calculate the magnetic helicity via Equation 4.1, we split the process into a few distinct steps. The first concern is that, to calculate $A \cdot B$, these two fields must be defined in the same locations. That is the only way to accurately multiply the components of the two fields. Because of the staggered nature of our grids (see Section 2.2.1), the first step is to interpolate the fields so that the components of $A$ and $B$ are colocated. After that, we can calculate the dot product in a very straightforward way. The final step is to perform the volume integral, which can be done in different ways. After calculating the helicity, we would also like a way to evaluate the values, which we will do using the corresponding values of helicity for a fully helical field. We treat each step below.

4.2.1 Interpolation of the Fields

To obtain representations of $A$ and $B$ at the same locations, we must interpolate at least one of the fields. To determine which field or fields need to be interpolated, it is necessary to decide where we want to calculate the dot product. To avoid giving preference to either field, we decide to calculate the dot product at either cell centers or cell vertices. In addition, generally scalars are defined at centers ($\rho, P, T$) or vertices ($\nabla \cdot A$), while vectors are defined on faces ($B$) or edges ($A$). The dot product is a scalar, so one of these locations is also natural in this regard.

We try both locations to see how this choice affects the overall helicity calculation. The interpolation of the fields to centers is considered first. For each component of the magnetic field, we need to average the values on the two faces of the cell along the relevant direction. That is, we perform the calculation

$$B_{c}^{i}(i, j, k) = \frac{1}{2} \left[ B^{i}(i - 1/2, j, k) + B^{i}(i + 1/2, j, k) \right],$$

(4.2)

where the subscript “c” stands for “center”, and there are similar expressions for $B_{c}^{j}$ and $B_{c}^{k}$. For the components of the vector potential, there are four edges along a given direction that must be
Chapter 4. Application to Magnetic Helicity

considered to get the value at the center of the cell. Thus the equation becomes

\[
A^i_C(i, j, k) = \frac{1}{4} \left[ A^i(i, j - 1/2, k - 1/2) + A^i(i, j - 1/2, k + 1/2) \\
+ A^i(i, j + 1/2, k - 1/2) + A^i(i, j + 1/2, k + 1/2) \right],
\]

(4.3)

again with equivalent expressions for \(A^j_C\) and \(A^k_C\). It should be noted that this calculation is equivalent to averaging to get \(A\) on the faces and then averaging again to get values at cell centers. For this method, the size of both \(A_C\) and \(B_C\) is \(L \times M \times N\), the same as the dimensions of the grid itself.

The other location at which we could calculate the dot product of \(A\) and \(B\) is at the vertices of cells. In this case, there are now four values of each component of \(B\) around a vertex and only two values of each component of \(A\). Therefore, in this scheme, Equation 4.2 becomes

\[
B^i_V(i + 1/2, j + 1/2, k + 1/2) = \frac{1}{4} \left[ B^i(i + 1/2, j, k) + B^i(i + 1/2, j, k + 1) \\
+ B^i(i + 1/2, j, k + 1) + B^i(i + 1/2, j + 1, k + 1) \right],
\]

(4.4)

where here the subscript “v” stands for “vertex,” with similar expressions for \(B^j_V\) and \(B^k_V\). Meanwhile, Equation 4.3 becomes

\[
A^i_V(i + 1/2, j + 1/2, k + 1/2) = \frac{1}{2} \left[ A^i(i, j + 1/2, k + 1/2) + A^i(i + 1, j + 1/2, k + 1/2) \right],
\]

(4.5)

with the corresponding expressions for \(A^j_V\) and \(A^k_V\). The sizes of \(A_V\) and \(B_V\) are a little different; because these calculations require the existence of values of \(A\) and \(B\) on either side of a vertex, we cannot calculate these fields for any vertex on the edge of the grid. Thus we lose two vertices along each direction, one on each side of the domain. The total number of vertices is \((L + 1) \times (M + 1) \times (N + 1)\), so the sizes of \(A_V\) and \(B_V\) are \((L - 1) \times (M - 1) \times (N - 1)\).

While there was no \textit{a priori} reason to favor one of these methods over the other, it does appear that by interpolating \(A\) and \(B\) to the vertices, we might lose some information. It is still interesting to see how this choice affects the magnetic helicity.
4.2.2 Computation of the Dot Product

Compared to the previous step, the calculation of the dot product is rather simple. Regardless of where we have colocated our fields, the computation is the same:

\[
\mathbf{A} \cdot \mathbf{B} = A^i B^i + A^j B^j + A^k B^k. \tag{4.6}
\]

The only differences are the locations at which \(\mathbf{A} \cdot \mathbf{B}\) is defined and its size, which match the corresponding locations and sizes of the \(\mathbf{A}\) and \(\mathbf{B}\) fields used to calculate it.

4.2.3 Evaluation of the Integral

Now that we have calculated the dot product of \(\mathbf{A}\) and \(\mathbf{B}\), the last step is to perform the integration to calculate the helicity (Equation 4.1). There are a myriad of ways to perform numerical integration. We begin with the simplest method: add up the value of the function at every point in the domain. When we interpolate the fields to cell centers, this takes the form

\[
H = \int_V (\mathbf{A} \cdot \mathbf{B})_c \, dV = dx \, dy \, dz \sum_{i=0}^{L-1} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} (\mathbf{A} \cdot \mathbf{B})_c(i, j, k) \tag{4.7},
\]

while for fields interpolated to vertices, we get

\[
H = \int_V (\mathbf{A} \cdot \mathbf{B})_v \, dV = dx \, dy \, dz \sum_{i=0}^{L-2} \sum_{j=0}^{M-2} \sum_{k=0}^{N-2} (\mathbf{A} \cdot \mathbf{B})_v(i + 1/2, j + 1/2, k + 1/2) \tag{4.8}.
\]

Another option is to use Simpson’s Rule to perform the integration. Simpson’s Rule states that for a function \(f(x)\) defined at points \((x_0, x_1, \ldots, x_n)\) with uniform separation \(h = x_{i+1} - x_i\), one way to perform an integration of \(f(x)\) over a pair of intervals

\[
\int_{x_{i-1}}^{x_{i+1}} f(x) \, dx = \frac{h}{3} \left[ f(x_{i-1}) + 4f(x_i) + f(x_{i+1}) \right]. \tag{4.9}
\]

When there are many pairs of intervals, we can get the overall integral by adding the integrals over
individual pairs of intervals, which results in
\[
\int_{x_0}^{x_n} f(x) \, dx = \frac{\hbar}{3} \left[ f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 4f(x_{n-1}) + f(x_n) \right].
\] (4.10)

To be useful for our problem, we must generalize this formula to three dimensions, which can be accomplished by simply using this method successively for each direction, just like evaluating any other nested integrals.

### 4.2.4 Fully Helical Fields

To aid in the understanding of the values of helicity we obtain by the methods described above, we would like benchmark values. One quantity against which we can compare is the helicity of a fully helical field. If a field is fully helical, the value of helicity is given by [52]
\[
H \sim \frac{B^2}{k},
\] (4.11)
where \( k \) is the characteristic wavenumber, given by
\[
k = \frac{2\pi}{\lambda}. \tag{4.12}
\]

For our purposes, we will set the wavelength \( \lambda \) equal to the size of our system, i.e., the size of the rotor or the neutron star. By comparing the values of helicity for the actual magnetic field configuration to what the helicity would be if the field were fully helical, we can get a better idea of how helical our field configurations are.

### 4.3 Helicity in GRMHD Simulations

We have calculated the helicity for the simulations used in Chapter 3 to test our inverse curl methods. At minimum, we performed the calculation at the initial time, the checkpoint time, and the final time. When the full data for \( \mathbf{B} \) were available at other times, we calculated the helicity at those times as well. It should be noted that in this case, the use of our inverse curl methods is
not necessary, as IllinoisGRMHD evolves the vector potential and therefore has access to both fields necessary to calculate helicity. It is still an illustrative example of how our methods can aid in the calculation of magnetic helicity.

4.3.1 Rotor Test

For the rotor test, as described in Section 3.2, we calculated the magnetic helicity using both of our definitions. The data are shown in Table 4.1. For the restarted run, we evaluated the helicity at the beginning of the simulation \((t = 0)\), the checkpoint time \((t = t_c)\), and the final time \((t = t_f)\). Additionally, at the checkpoint time we perform the calculation both before and after the inverse curl operation is performed, and we use the cell-by-cell method to evaluate the inverse curl.

Note that at the initial time, and at all times before \(t_c\), the two runs are identical. Even after our inverse curl technique is applied, but before the evolution continues, the value of the helicity has not changed, as shown by the identical values of \(t_{c,\text{before}}\) and \(t_{c,\text{after}}\). It is only after the restart that the two runs differ, so we only give the helicity for the final time of the uninterrupted run for comparison. We find that the helicity does appear to evolve over time, although the values are, for the most part, on an order of \(10^{-4} - 10^{-5}\), so it is not clear if the helicity is actually evolving or if it is so small that the code cannot track it properly. There are other possibilities as well, such as truncation error.\(^1\) There is also good agreement between the restarted run and the uninterrupted run, giving another indication that our methods give fields that can be evolved, assuming we can trust these values for helicity. In addition, we find that the values are orders of magnitude smaller than those for a fully helical field, meaning the fields in this test are not extremely helical.

4.3.2 Magnetized Neutron Stars

For the NS tests, as described in Section 3.3, we again calculated the magnetic helicity using both of our definitions. It should be noted that because the magnetic fields are set up to be completely poloidal, we can reasonably expect the helicity to be zero, at least at the beginning of the simulation. For the restarted run, we evaluated the helicity at the beginning of the simulation

\(^1\)See Section 4.3.2.1 for a discussion of truncation error in the stable NS test.
Chapter 4. Application to Magnetic Helicity

<table>
<thead>
<tr>
<th>Time</th>
<th>Magnetic Helicity (Rotor)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cell Centers</td>
<td>Cell Vertices</td>
<td>Fully Helical</td>
</tr>
<tr>
<td>0.0</td>
<td>$-1.4505 \times 10^{-04}$</td>
<td>0.0000</td>
<td>$\sim 1 \times 10^{-02}$</td>
</tr>
<tr>
<td>$t_c$, before</td>
<td>$-1.8990 \times 10^{-04}$</td>
<td>$-3.1919 \times 10^{-05}$</td>
<td>$\sim 7 \times 10^{-02}$</td>
</tr>
<tr>
<td>$t_c$, after</td>
<td>$-1.8990 \times 10^{-04}$</td>
<td>$-3.1919 \times 10^{-05}$</td>
<td>$\sim 7 \times 10^{-02}$</td>
</tr>
<tr>
<td>$t_f$</td>
<td>$-1.3275 \times 10^{-04}$</td>
<td>$-6.6931 \times 10^{-05}$</td>
<td>$\sim 1 \times 10^{-01}$</td>
</tr>
<tr>
<td>$t_f$, uni</td>
<td>$-1.3259 \times 10^{-04}$</td>
<td>$-6.6620 \times 10^{-05}$</td>
<td>$\sim 1 \times 10^{-01}$</td>
</tr>
</tbody>
</table>

Table 4.1: Values of the magnetic helicity for the rotor test. All the quantities are from the restarted run, except for those at $t_f$, uni, which are quantities from the final time of the uninterrupted run. The subscripts “before” and “after” are relative to the inverse curl operation done at the checkpoint time of the restarted run. The last column shows what the value of the helicity would be if the field configuration was fully helical.

For the interrupted run, we only evaluated the helicity at the final time ($t_f$), the checkpoint time ($t = t_{dyn}$), and the final time ($t = 4t_{dyn}$). Again we calculate the helicity both before and after the inverse curl operation is performed at the checkpoint time, and again we use the cell-by-cell method to evaluate the inverse curl. For the uninterrupted run, we again only evaluated the helicity at the final time, for the same reason as in the rotor test. The results for each type of NS we evolved are given in the next two sections.

### 4.3.2.1 Stable Neutron Stars

The values of helicity for the stable NS are given in Table 4.2. Here again, we only give the helicity for the final time of the uninterrupted run for comparison. We find that the helicity is consistent with zero through the whole simulation. The helicity values here also match before and after the inverse curl operation is performed in the restarted run, and we still see good agreement between the restarted run and the uninterrupted run. Here the values are only one order of magnitude smaller than those for a fully helical field, meaning the fields in this test are more helical than those for the rotor test.

Another test we can do to check whether the evolution of the helicity is actually physical or just a numerical artifact is a convergence test. As we increase the number of gridpoints, what happens to the value of the helicity? In Table 4.3, we show values of helicity for the stable NS at the checkpoint time, $t_{dyn}$, before the inverse curl operation is performed, which is $t_{dyn}$, before. These resolutions are the same ones used for the performance tests in Section 3.4, where $84^3$ is
Chapter 4. Application to Magnetic Helicity

### 4.3.2.2 Unstable Neutron Stars

Table 4.4 shows the helicity data for the unstable NS. While the helicity here does not get quite as small as in the stable case, it is still fairly consistent with zero. Even so, the helicity values are still the same before and after the inverse curl operation is performed in the restarted run. Here the restarted run and the uninterrupted run do not match each other quite as well as in the other tests, but the magnetic field configuration of this test is less well-behaved, so this is not very surprising.
As in the stable case, the helicity values are only one order of magnitude smaller than those for a fully helical field, meaning the fields in this test are just as helical as those for the stable NS.

<table>
<thead>
<tr>
<th>Time</th>
<th>Magnetic Helicity (Unstable NS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cell Centers</td>
</tr>
<tr>
<td>0.0</td>
<td>5.1179 × 10⁻⁶</td>
</tr>
<tr>
<td>t_{dyn,before}</td>
<td>4.8159 × 10⁻⁵</td>
</tr>
<tr>
<td>t_{dyn,after}</td>
<td>4.8159 × 10⁻⁵</td>
</tr>
<tr>
<td>4t_{dyn}</td>
<td>−7.5075 × 10⁻⁵</td>
</tr>
<tr>
<td>4t_{dyn,uni}</td>
<td>−7.2964 × 10⁻⁵</td>
</tr>
</tbody>
</table>

Table 4.4: Values of the magnetic helicity for the unstable neutron star test. Once again all the quantities are from the restarted run, except for those at 4t_{dyn,uni}, which are quantities from the final time of the uninterrupted run. The subscripts “before” and “after” are still relative to the inverse curl operation done at the checkpoint time of the restarted run. The last column shows what the value of the helicity would be if the field configuration was fully helical.

4.4 Summary

Magnetic helicity is a quantity that can give information about the magnetic field configuration. Because it is defined as the integral over \( \mathbf{A} \cdot \mathbf{B} \), it can be difficult to calculate if the only quantity available is the magnetic field. One situation in which this is the case is a GRMHD simulation using codes like HARM3D that directly evolve the magnetic field, as opposed to codes like ILLINOISGRMHD that evolve the vector potential.

Another concern with calculating helicity for a code such as ILLINOISGRMHD is the staggered nature of the grid. Before calculating \( \mathbf{A} \cdot \mathbf{B} \), we must first transform the \( \mathbf{A} \) and \( \mathbf{B} \) fields so that they are colocated.

By calculating the helicity at different times within a single simulation, we can determine if the helicity is conserved, as it should be. For the tests we have already done, we find very small values of the helicity: on the order of \( 10^{-5} - 10^{-4} \) for the rotor test, \( 10^{-10} - 10^{-9} \) for the stable NS, and \( 10^{-6} - 10^{-5} \) for the unstable NS. While the values fluctuate over the course of a simulation, they are so small that this “evolution” is possibly spurious, caused by numerical error, meaning that the helicity is conserved. We also see that the inverse curl operation itself does not result in
any changes to the helicity whatsoever, which is a good sign for our methods. In addition we find good agreement between our restarted runs and the uninterrupted runs, lending more validation to our inverse curl methods. Finally, comparing to values of helicity corresponding to fully helical configuration, we find that none of our tests result in fully helical fields, but the NS tests come closer than the rotor test.
CHAPTER 5

DISCUSSION

5.1 Summary

The techniques described here are sufficient to construct a vector potential on a staggered numerical grid, appropriate for use in applications in which magnetic fields evolved using, e.g., a constrained transport approach, must be converted into a vector potential. Our solution consists of a numerical implementation of the “inverse curl” operator over a finite rectangular domain. This problem is typically much more difficult to solve than the case of a spherical domain, for which a spectral solution can be written down explicitly in terms of vector spherical harmonic modes. Additionally, these methods can be applied to any problem in which two quantities are related by the curl, such as the fluid velocity and vorticity in the context of fluid dynamics.

Our first technique is a cell-by-cell generation scheme, in which the six values of $B$ for a given cell are used to generate the 12 values of $A$ for that cell. As we move from cell to cell in the grid, we need to ensure consistency between cells because most cell edges, where $A$ is defined, are shared by multiple cells. Additionally, this method gives an $A$ field in an arbitrary gauge; we perform a gauge transformation to put the field in the Coulomb gauge.

The other technique we have developed is a global linear algebra method, which treats the curl operator as a very large square matrix that acts on a column vector containing all the values of all
three components of $A$ to give all the values of all three components of $B$. The matrix is sparse, but not diagonally dominant. As such, we can use standard linear algebra packages to solve for vector potential.

We performed various evolution tests on our methods to see if they can actually be used in GRMHD simulations. The rotor test and evolutions of neutron stars, both weakly magnetized and strongly magnetized, demonstrate that the vector potentials output by our methods can in fact be fed back into ILLINOISGRMHD to continue simulations and still result in the same magnetic fields at the ends of the simulations, compared with baseline simulations.

Performance testing shows that each of our techniques possesses unique strengths and weaknesses. The cell-by-cell method is very fast and scales as $O(N)$ in both time and memory. On the other hand, the global linear algebra method requires more memory and is slower, but it is a much more symmetric technique that uniformly cleans nonzero divergences in the input magnetic fields and applies Coulomb gauge conditions as it solves, instead of as a separate step. The global linear algebra method is also much more amenable to the implementation of different boundary conditions and mesh refinement (discussed below) than the cell-by-cell method.

Another application of our methods is to the calculation of magnetic helicity, which requires both the vector potential and the field itself to be known. With only the magnetic field, one could use our inverse curl methods to get the vector potential and calculate the helicity. By calculating the helicity at different times throughout the simulations used to test our methods, we find that for all our tests, the helicity is small and does not vary much, indicating that it is indeed conserved, at least to an approximate level. The agreement between helicity values calculated from our restarted runs and uninterrupted runs also gives another boost to the validity of our methods. We also compared the calculated values of helicity to what the values would be for a fully helical field and find that, while none of our tests results in fully helical fields, the NS tests result in fields that are closer to being fully helical.
5.2 Future Work

There are always ways to continue to improve and optimize any numerical code, and our methods are no exception. One improvement we are already considering is more flexible input and output subroutines. This will allow more people to use our methods with minimal alteration. Of course, as the codes are open-source through GitHub [54], anyone can download the codes and adapt them however they want. An additional possible avenue of improvement is to parallelize the cell-by-cell method. It is a very fast method already, but it is always useful to have the option of running in parallel.

Another path forward is to implement mesh refinement, which is an important tool used in numerical relativity simulations to apply additional resolution in regions where it is necessary, such as near BHs or NSs. Both HARM3D and ILLINOISGRMHD have the capability to use mesh refinement, and so any method aimed at bridging the gap between them via inverse curl algorithms must also have such capabilities. Unfortunately, the staggering of the grid in the cell-by-cell method makes mesh refinement, even by a simple factor of two, prohibitively complicated and most likely infeasible. The global linear algebra method, however, can handle mesh refinement with extremely careful consideration. Another way to implement mesh refinement is to use an inherently multi-grid method, which can be explored in the future.

In addition to improving our methods, we can also continue applying them to interesting problems involving magnetic phenomena. The most natural immediate goal would be to continue investigating the magnetic helicity in GRMHD simulations. In particular, it would be useful to calculate the helicity in HARM3D simulations, as HARM3D cannot natively calculate helicity at the moment.

Then there is the motivating problem of this work: performing a full simulation of binary BHs, or other systems involving compact objects, from initial inspiral through merger and ringdown. More specifically, we want to perform the early stages of such a simulation in HARM3D and then finish it in ILLINOISGRMHD, which relies on the inverse curl to accurately model the magnetic effects across the transition between codes.
5.3 Acknowledgments

This work was supported in part by NSF award OAC-1550436. Computational resources were provided by the NewHorizons cluster at RIT, which was supported by NSF grants No. PHY-0722703, No. DMS-0820923, and No. AST-1028087.

Collaborators Zachariah Etienne, Ian Ruchlin, and Thomas Adams were supported in part by NSF EPSCoR Grant OIA-1458952. Some early MUMPS computations were performed on WVU’s Spruce Knob high-performance computing cluster, funded in part by NSF EPSCoR Research Infrastructure Improvement Cooperative Agreement #1003907, the state of West Virginia (WVEPSCoR via the Higher Education Policy Commission), and West Virginia University.


URL https://doi.org/10.12942/lrr-2012-8

URL https://doi.org/10.12942/lrr-2011-6

URL http://stacks.iop.org/1742-6596/120/i=3/a=032007


[54] GitHub repository containing our methods.
URL https://github.com/zsilberman/Inverse-Curl

URL http://eigen.tuxfamily.org


URL https://doi.org/10.1137/S0895479899358194

