

Topics in Numerical Relativity: Horizon Finding, Asynchronous Parallelism, and Cross-Code Comparisons of Gravitational Waveforms

Thesis by
Kyle Christopher Nelli

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended May 15, 2026

© 2026

Kyle Christopher Nelli
ORCID: 0000-0003-2426-8768

All rights reserved

*It should just work.*TM

ACKNOWLEDGEMENTS

To my advisors who guided and mentored me throughout the years: Thank you **Saul Teukolsky** for giving me the freedom to explore the numerical side of numerical relativity, which has grown into a passion for software engineering. Thank you **Mark Scheel** for the long conversations about everything numerical relativity related early in my PhD and the (often headache riddled) chats about control systems. I wouldn't be the researcher or scientist I am today without you two.

To additional mentors who eventually became friends somewhere along the way: Thank you **Nils Deppe** for taking me under your wing during the early years of my PhD, teaching me about both the beauty and horror that C++ can be, and generally inspiring my interest in software engineering. Thank you **Nils Vu** for preserving the conservation-of-Nils and being there to help guide me in the latter part of my PhD. I am honored to help bookend your time at Caltech by picking you, **Trang**, and **Charlie** up and dropping you off at the airport. And thank you to **Geoffrey Lovelace** for welcoming me at Cal State Fullerton and sharing many stories of your time at Caltech as a PhD.

To my collaborators and friends around the world who I have thoroughly enjoyed meeting and working with: Thank you **Marcie Bonilla**, **Alex Carpenter**, **Deborah Ferguson**, **Francois Foucart**, **Noora Ghadiri**, **Larry Kidder**, **Alex Macedo**, **Tia Martineau**, **PJ Nee**, **Mike Pajkos**, **Harald Pfeiffer**, **Leo Stein**, **Will Throwe**, **Niko Wittek**, **Hengrui Zhu**, and many more.

To the members of Saul's research group, both past and present, who enhanced my time at Caltech by making the good days great and the bad days ok, who know the struggle of numerical relativity all too well, and who never seem to make their axis labels large enough in their presentations: Thank you **Keefe Mitman**, **Sizheng Ma**, **Yoonsoo Kim**, **Sarah Habib**, **Dongzu Sun**, **Himanshu Chaudhary**, **Isaac Legred**, **Isabella Pretto**, **Talya Klinger**, **Guido Da Re**, **Taylor Knapp**, **Vittoria Tommasini**, and **Iago Mendes**.

To a TAPIR legend: Thank you **JoAnn Boyd** for always complimenting me on my expense reports. They're honestly not that hard.

To my closest friends: Thank you **Carla Stelsel** and **Hannah Manetsch** for all the support, encouragement, and advice you have given me since our time together in undergrad, especially when I did not make it easy. The depth of your kindness and

compassion cannot be overstated and will never be forgotten. Thank you **Radhika Bhatt** for our conversations in the kitchen about both work and life. You made our home a comforting and safe space for me during difficult years, for which I am forever grateful.

To all my friends, whether at Caltech or not, who supported me and made a grueling ordeal tolerable: Thank you **David Abramovitch, Sasha Alabugin, Ian Foo, Elijah Kane, Quinn Morgan, Katelyn Mulcahy, Nicholas Rui, Kiran Shila, Rhiannon Udall, Eli Weissler, Ethan Wolfe**, all of my friends at UAW 2478, and so many more.

To my minivan, TRINKT2: Thank you for taking me across the country seven times during my PhD and for helping me and countless friends move couches and apartments. Your memory will live on in TRINKT3.

And finally, to my family without whom none of this would be possible: Thank you to my parents **Trish & Chris Nelli**, my sister **Erin**, and our now departed furry friend **Rylee** for all the love and support you have given me while I've been off at both undergrad and grad school. Thank you for always welcoming me home with open arms and a Chicago staple, either Portillo's or Giordano's. And thank you for attempting to understand what I study, even if you may not know or care. I truly could not have done this without you.

ABSTRACT

From their prediction in 1916 by Einstein, to their first detection almost 100 years later in 2015, gravitational waves provide a new way of looking at the universe, completely separate from electromagnetic radiation. They allow us to probe and study the most extreme gravitational environments that our universe has to offer: black holes. When two black holes merge, they produce a burst of gravitational waves which, if energetic enough, are able to be detected here on Earth.

However, to effectively model these mergers, we cannot solely rely on analytical descriptions. They break down near the actual merger, when spacetime is at its most extreme. To model this merger, we need numerical simulations of mergers to predict what they will look like. These simulations must be extremely accurate due to the incredible precision with which we are able to detect gravitational waves. Therefore, these expensive simulations need to run on supercomputers and be highly efficient in order to meet the necessary accuracy requirements.

In this thesis, we present a novel algorithm for efficiently finding the apparent horizons in numerical simulations done with task-based parallelism. The performance improvement of task-based parallelism over more traditional parallelism is significant, but it introduces considerable algorithmic complexity which has to be dealt with. We use this novel algorithm to perform the first binary black hole merger done with discontinuous Galerkin methods and analyze the waveform output. Lastly, we present a new feature to the SpECTRE CCE module which allows it to be run on data from other numerical relativity codes. We show this new feature working on simulation data from five different numerical relativity codes.

PUBLISHED CONTENT AND CONTRIBUTIONS

Chapter 2 [1]. KCN wrote the code, ran simulations, and wrote a majority of the manuscript.

Chapter 3 [2]. KCN wrote a majority of the code, helped debug simulations, and wrote a significant portion of the manuscript.

- [1] Kyle C. Nelli, William Throwe, Nils Deppe, Mark A. Scheel, Lawrence E. Kidder, Nils L. Vu, and Saul A. Teukolsky. Horizon tracking for asynchronous parallel black hole simulations. August 2025. URL <https://arxiv.org/abs/2508.08408>.
- [2] Geoffrey Lovelace, Kyle C. Nelli, et al. Simulating binary black hole mergers using discontinuous Galerkin methods. *Class. Quant. Grav.*, 42(3):035001, 2025. doi: 10.1088/1361-6382/ad9f19.

OTHER CO-AUTHOR CONTRIBUTIONS

Major contributions: [1], [2], [3]

SpECTRE core developer and maintainer of Cauchy-Characteristic Extraction (CCE) module: [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]

Bibliography

- [1] Nils Deppe et al. Binary neutron star mergers using a discontinuous Galerkin-finite difference hybrid method. *Class. Quant. Grav.*, 41(24):245002, 2024. doi: 10.1088/1361-6382/ad88cf.
- [2] Sizheng Ma et al. Fully relativistic three-dimensional Cauchy-characteristic matching for physical degrees of freedom. *Phys. Rev. D*, 109(12):124027, 2024. doi: 10.1103/PhysRevD.109.124027.
- [3] Milton Ruiz, Antonios Tsokaros, Stuart L. Shapiro, Kyle C. Nelli, and Sam Qunell. Magnetic ergostars, jet formation and gamma-ray bursts: Ergoregions versus horizons. *Phys. Rev. D*, 102(10):104022, 2020. doi: 10.1103/PhysRevD.102.104022.
- [4] Jiayi Wu, Elias R. Most, Nils L. Vu, Nils Deppe, Lawrence E. Kidder, Kyle C. Nelli, and William Throwe. Eccentricity as a signature of hierarchical subsolar-mass mergers in collapsar disks. April 2026. URL <https://arxiv.org/abs/2604.26912>.
- [5] Aniket Khairnar, Leo C. Stein, Michael Boyle, Nils Deppe, Lawrence E. Kidder, Keefe Mitman, Jordan Moxon, Kyle C. Nelli, William Throwe, and Nils L. Vu. Fixing the center-of-mass frame of numerical relativity waveforms using the post-Newtonian center-of-mass charge. March 2026. URL <https://arxiv.org/abs/2603.24661>.
- [6] Sarah Habib et al. Error quantification and comparison of binary neutron star gravitational waveforms from numerical relativity codes. September 2025. URL <https://arxiv.org/abs/2509.23028>.
- [7] Guillermo Lara et al. Signatures from metastable oppositely-charged black hole binaries in scalar Gauss-Bonnet gravity. May 2025. URL <https://arxiv.org/abs/2505.14785>.
- [8] Mark A. Scheel et al. The SXS collaboration’s third catalog of binary black hole simulations. *Class. Quant. Grav.*, 42(19):195017, 2025. doi: 10.1088/1361-6382/adfd34.

- [9] Keefe Mitman et al. Probing the ringdown perturbation in binary black hole coalescences with an improved quasinormal mode extraction algorithm. *Phys. Rev. D*, 112(6):064016, 2025. doi: 10.1103/qq1g-jlnw.
- [10] Guido Da Re et al. Modeling the BMS transformation induced by a binary black hole merger. *Phys. Rev. D*, 111(12):124019, 2025. doi: 10.1103/PhysRevD.111.124019.
- [11] Nils Deppe, Lavinia Heisenberg, Lawrence E. Kidder, David Maibach, Sizheng Ma, Jordan Moxon, Kyle C. Nelli, William Throwe, and Nils L. Vu. Signatures of quantum gravity in gravitational wave memory. *Phys. Rev. D*, 112(2):024016, 2025. doi: 10.1103/7c2f-975v.
- [12] Leda Gao et al. Robustness of extracting quasinormal mode information from black hole merger simulations. *Phys. Rev. D*, 112(2):024025, 2025. doi: 10.1103/3jj6-jc8q.
- [13] Sizheng Ma, Mark A. Scheel, Jordan Moxon, Kyle C. Nelli, Nils Deppe, Lawrence E. Kidder, William Throwe, and Nils L. Vu. Merging black holes with Cauchy-characteristic matching: Computation of late-time tails. *Phys. Rev. D*, 112(2):024003, 2025. doi: 10.1103/jd26-8q5w.
- [14] Marina De Amicis et al. Late-time tails in nonlinear evolutions of merging black holes. *Phys. Rev. Lett.*, 135(17):171401, 2025. doi: 10.1103/2brx-xnyr.
- [15] Matthew Giesler et al. Overtones and nonlinearities in binary black hole ringdowns. *Phys. Rev. D*, 111(8):084041, 2025. doi: 10.1103/PhysRevD.111.084041.
- [16] Nils Deppe, Lavinia Heisenberg, Henri Inchauspé, Lawrence E. Kidder, David Maibach, Sizheng Ma, Jordan Moxon, Kyle C. Nelli, William Throwe, and Nils L. Vu. Echoes from beyond: Detecting gravitational-wave quantum imprints with LISA. *Phys. Rev. D*, 111(12):124035, 2025. doi: 10.1103/k7jh-rhgw.
- [17] Nikolas A. Wittek, Leor Barack, Harald P. Pfeiffer, Adam Pound, Nils Deppe, Lawrence E. Kidder, Alexandra Macedo, Kyle C. Nelli, William Throwe, and Nils L. Vu. Relieving scale disparity in binary black hole simulations. October 2024. URL <https://arxiv.org/abs/2410.22290>.
- [18] Sizheng Ma, Kyle C. Nelli, Jordan Moxon, Mark A. Scheel, Nils Deppe, Lawrence E. Kidder, William Throwe, and Nils L. Vu. Einstein–Klein–Gordon system via Cauchy-characteristic evolution: Computation of memory and ringdown tail. *Class. Quant. Grav.*, 42(5):055006, 2025. doi: 10.1088/1361-6382/adaf6f.
- [19] Lorena Magaña Zertuche et al. High-precision ringdown surrogate model for nonprecessing binary black holes. *Phys. Rev. D*, 112(2):024077, 2025. doi: 10.1103/q7sy-g3kl.

- [20] Teagan A. Clarke et al. Toward a self-consistent framework for measuring black hole ringdowns. *Phys. Rev. D*, 109(12):124030, 2024. doi: 10.1103/PhysRevD.109.124030.
- [21] Yitian Chen et al. Improved frequency spectra of gravitational waves with memory in a binary-black-hole simulation. *Phys. Rev. D*, 110(6):064049, 2024. doi: 10.1103/PhysRevD.110.064049.
- [22] Guillermo Lara, Harald P. Pfeiffer, Nikolas A. Wittek, Nils L. Vu, Kyle C. Nelli, Alexander Carpenter, Geoffrey Lovelace, Mark A. Scheel, and William Thorne. Scalarization of isolated black holes in scalar Gauss-Bonnet theory in the fixing-the-equations approach. *Phys. Rev. D*, 110(2):024033, 2024. doi: 10.1103/PhysRevD.110.024033.
- [23] Hengrui Zhu et al. Nonlinear effects in black hole ringdown from scattering experiments: Spin and initial data dependence of quadratic mode coupling. *Phys. Rev. D*, 109(10):104050, 2024. doi: 10.1103/PhysRevD.109.104050.
- [24] Hengrui Zhu et al. Black hole spectroscopy for precessing binary black hole coalescences. *Phys. Rev. D*, 111(6):064052, 2025. doi: 10.1103/PhysRevD.111.064052.
- [25] Nils Deppe et al. Simulating magnetized neutron stars with discontinuous Galerkin methods. *Phys. Rev. D*, 105(12):123031, 2022. doi: 10.1103/PhysRevD.105.123031.

TABLE OF CONTENTS

Acknowledgements	iv
Abstract	vi
Published Content and Contributions	vii
Other Co-Author Contributions	viii
Bibliography	viii
Table of Contents	xi
List of Illustrations	xiii
List of Tables	xv
Nomenclature	xvi
Chapter I: Introduction	1
1.1 Newtonian Gravity	1
1.2 Special Relativity	3
1.3 General Relativity	5
1.4 Numerical Relativity	12
1.4.1 BSSN Formulation	15
1.4.2 Generalized Harmonic Formulation	19
1.5 Goals of NR	26
1.6 Why HPC is Necessary	27
1.7 Horizon Finding in the GH Formalism	29
1.8 Overview of Chapters	32
1.8.1 Chapter 2	32
1.8.2 Chapter 3	32
1.8.3 Chapter 4	32
Bibliography	33
Chapter II: Horizon tracking for asynchronous parallel black hole simulations	39
2.1 Introduction	40
2.2 Asynchronous Parallelism	42
2.2.1 Difference from Synchronous Parallelism	42
2.2.2 I/O Example	43
2.2.3 Complications with I/O Example	44
2.3 Apparent Horizon Finder	45
2.4 Synchronous Control System	49
2.4.1 Computational Domain	50
2.4.2 Time-Dependent Coordinate Mapping	51
2.4.3 Control Error and Signal	53
2.4.4 Computing $\lambda^{\text{new}}(t)$	55
2.5 Asynchronous Control System	56
2.5.1 Global State	56
2.5.2 Expiration Time	57

2.5.3	Querying $\lambda(t)$ and Computing $\lambda^{\text{new}}(t)$	60
2.5.4	Eventual Concurrency	62
2.5.5	Summary	63
2.5.6	Deadlocks and Errors	64
2.6	Timing Results	65
2.7	Application to Binary Neutron Stars	66
2.8	Conclusion	68
	Bibliography	69
	Chapter III: Simulating binary black hole mergers using discontinuous Galerkin methods	76
3.1	Introduction	77
3.2	Methods	79
3.2.1	Equations of Motion	79
3.2.2	Discontinuous Galerkin (DG) Method	81
3.2.3	Boundary Conditions	84
3.2.4	Spectral Filter	86
3.2.5	Time Integration	86
3.2.6	Gauge Condition	86
3.2.7	Control Systems	87
3.3	Results	88
3.3.1	Single Black Hole Evolutions	88
3.3.2	Gauge Wave	91
3.3.3	Binary Black Hole Inspiral, Merger, and Ringdown	93
3.4	Conclusion	107
	Bibliography	109
	Chapter IV: Infrastructure for cross-code comparison of CCE waveforms . . .	120
4.1	Introduction	120
4.1.1	Electricity and Magnetism Example	120
4.1.2	General and Numerical Relativity Gauges	121
4.2	Cauchy-Characteristic Extraction	124
4.2.1	Characteristic Evolution	125
4.2.2	Frame Fixing and BMS Transformations	127
4.3	Worldtube Data Formats	128
4.3.1	Original Worldtube Data Format	128
4.3.2	New Worldtube Data Formats	129
4.4	Examples of NR Codes Using SpECTRE CCE	131
4.4.1	SpECTRE	131
4.4.2	SpEC	133
4.4.3	AthenaK	134
4.4.4	FIL	134
4.4.5	Einstein Toolkit	137
4.5	Conclusion	137
	Bibliography	140

LIST OF ILLUSTRATIONS

<i>Number</i>	<i>Page</i>
1.1 Light cone centered at event P	4
1.2 Parallel transport on a flat circle.	6
1.3 Parallel transport on a sphere.	7
1.4 Foliation of spacetime into spatial hypersurfaces.	13
1.5 Lapse and shift in the 3+1 decomposition.	14
1.6 Inspiral, merger, and ringdown diagram.	27
1.7 Null normals to a closed 2-surface (apparent horizon).	30
2.1 Diagram of trial horizon surfaces moving between cells during fast-flow.	47
2.2 Equatorial slices of excised spherical domains for single black hole evolutions.	51
2.3 Timeline comparing synchronous and asynchronous computation of $\lambda^{\text{new}}(t)$	58
2.4 Snapshots of rest mass density and grid for a BNS system.	67
3.1 Diagram of forward and reverse coordinate mappings for an element.	82
3.2 Computational domain used for single black hole evolutions.	89
3.3 Constraint violations for single black hole evolutions in Kerr-Schild coordinates.	90
3.4 Constraint violations for single black hole evolutions in harmonic coordinates.	91
3.5 Constraint violations for single black hole evolutions in damped har- monic coordinates.	92
3.6 Gauge wave simulation demonstrating long-term stability and con- vergence.	93
3.7 Apparent horizon trajectories during the BBH inspiral.	95
3.8 Computational grid used during the BBH inspiral.	96
3.9 Computational domain used during ringdown.	98
3.10 Constraint energy for equal-mass non-spinning BBH at three resolu- tions.	102
3.11 Apparent horizon masses and spins during inspiral and ringdown. . .	104
3.12 The $\ell = m = 2$ gravitational-wave mode computed via CCE at three resolutions.	106

4.1	SpECTRE GW modes mapped to the inspiral superrest frame at three resolutions.	132
4.2	SXS:BBH:2696 GW modes mapped to the inspiral superrest frame. .	133
4.3	GW150914-type GW modes mapped to the inspiral superrest frame. .	135
4.4	BNS GW modes from SpEC and FIL mapped to the inspiral superrest frame.	136
4.5	ETK equal-mass non-spinning BBH GW modes mapped to the inspiral superrest frame.	138

LIST OF TABLES

<i>Number</i>	<i>Page</i>
3.1 Parameters used to generate the BBH initial data.	95
3.2 Constraint damping parameters during the inspiral.	101
3.3 Constraint damping parameters during ringdown.	101
3.4 L2-norm of constraint energy and mapping error at $t = 3000M$	102

NOMENCLATURE

ADM	Arnowitt-Deser-Misner
AH	Apparent Horizon
BBH	Binary Black Hole
BH	Black Hole
BMS	Bondi–van der Burg–Metzner–Sachs
BNS	Binary Neutron Star
BSSN	Baumgarte-Shapiro-Shibata-Nakamura
BSSNOK	Baumgarte-Shapiro-Shibata-Nakamura-Oohara-Kojima
CCE	Cauchy-Characteristic Extraction
CPU	Central Processing Unit
DG	Discontinuous Galerkin
DH	Damped harmonic
ETK	Einstein Toolkit
FOGH	First-Order Generalized Harmonic
GH	Generalized Harmonic
GR	General Relativity
GRMHD	General Relativistic Magnetohydrodynamics
GPU	Graphics Processing Unit
GW	Gravitational Wave
HPC	High Performance Computing
KAGRA	Kamioka Gravitational-wave Detector
LIGO	Laser Interferometer Gravitational-Wave Observatory
LISA	Laser Interferometer Space Antenna
LTS	Local Time Stepping
MPI	Message Passing Interface

NR	Numerical Relativity
NS	Neutron Star
PM	Post-Minkowskian
PN	Post-Newtonian
PND	Proportional- N th-Derivative
SR	Special Relativity
SXS	Simulating eXtreme Spacetimes
XCTS	Extended Conformal Thin Sandwich
Z4	Z4 Formulation
Z4c	Z4 Formulation with Constraint Damping

Chapter 1

INTRODUCTION

The gravity that we know here on Earth is intuitive. It is familiar. And it is safe (for the most part). But this universe that we live in continues to offer so much more than we expected. Starting from observing the motion of the planets and stars in the night sky to theorizing about black holes, the most compact and extreme objects in the universe, gravity continues to humble us.

1.1 Newtonian Gravity

In order to explain black holes, the overarching topic of this thesis, we first start with a more simplified version of gravity and work our way up. Newtonian gravity states that the gravitational force between two masses is proportional to the two masses and inversely proportional to the square of the distance between the masses, or

$$\mathbf{F} = -\frac{Gm_1m_2}{r^2}\hat{\mathbf{r}}, \quad (1.1)$$

with G being Newton's constant. However, we usually don't just have point masses like m_1 and m_2 . We often have a distribution of matter which we'll represent as a matter density ρ . This matter distribution will generate a gravitational potential Φ via Poisson's equation

$$\nabla^2\Phi = 4\pi G\rho. \quad (1.2)$$

To relate Eqs. 1.1 and 1.2, we compute the gradient of Φ as a force,

$$\mathbf{F} = -\nabla\Phi. \quad (1.3)$$

Then, because of Newton's second law

$$\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{x}}{dt^2}, \quad (1.4)$$

we can get the equations of motion for any object in Newtonian gravity,

$$m\frac{d^2x_i}{dt^2} = -\frac{\partial\Phi}{\partial x_i}. \quad (1.5)$$

Additionally, there is a physical constraint on the matter density ρ , namely the conservation of mass. This looks like

$$\frac{\partial\rho}{\partial t} = -\nabla \cdot (\rho\mathbf{v}). \quad (1.6)$$

This equation states that the rate of change of matter over time is given by the flux of matter entering or leaving a region.

With these equations, one is able to compute quite a lot of gravitational scenarios. This could be the motion of a planet around a star or moving a satellite to a Lagrange point of two orbiting bodies. Combined with some ideal fluid mechanics for computing ρ , they can also be used to model the interiors of large gas planets or even stars. These basic starting equations are able to unlock a large amount of gravitational physics on their own, making them vital, even today, to astrophysical research.

However, Newtonian gravity does have its limitations. One is how simultaneity of events is measured. In order to understand simultaneous events, we must understand the coordinates that are used and how they transform in Newtonian gravity. If we take a step back, one way to conceptually think about Newtonian gravity is that we have a time coordinate t . And at each time t , we have a snapshot of our 3D world which is represented by some spatial coordinates x, y, z . Therefore, two events that happen simultaneously will be labeled with the same value for the time coordinate t , no matter where we are in our spatial coordinates. However, we have some freedom with our spatial coordinates. We can freely transform them from cartesian (x, y, z) to spherical (r, θ, ϕ) to cylindrical (s, ϕ, z) or to any other coordinates we can think of without altering our time coordinate, preserving simultaneity. Conversely, the only way to transform our time coordinate (and keep the same physics) is a simple $t \rightarrow t + \delta t$. Therefore, time and space are somewhat decoupled in Newtonian gravity.

Another limitation of Newtonian gravity is that it doesn't account for the finite speed of light. Related to the first limitation, events can only be considered simultaneous if they occur at the same time t . Consider a scenario where one person stands at rest at a point (x, y, z) and time t and shines a flashlight in the $+x$ direction. A second person is moving with respect to the first person in the $+x$ direction with a velocity v . This second person has their own coordinate system (t', x', y', z') different from the first person at rest and would find the relation between the two systems to be

$$t' = t \tag{1.7}$$

$$x' = x - vt \tag{1.8}$$

$$y' = y \tag{1.9}$$

$$z' = z. \tag{1.10}$$

Consider what happens if the second person is moving very close to the speed of light (i.e. $v \approx c$). One may be tempted to think of this scenario as similar to a different scenario where the first person is throwing a baseball while the second rides in a car. It'd be perfectly reasonable for a car to be moving faster than a baseball. However, this is fundamentally impossible with light. The speed of light is constant in all coordinate systems. So if we try to interpret this scenario in a Newtonian way, the second person would be moving almost in line with the front of the light beam at the same speed. But then what would happen if this second person shined their own flashlight? Would this second beam of light be moving almost twice as fast as the first one? Would it stay effectively stationary and not go anywhere because the second person was already moving at the speed of light? This inconsistency and impossibility is why Newtonian gravity isn't the full picture and why we need a different theory.

1.2 Special Relativity

The theory of special relativity (SR) is able to reconcile and account for the finite speed of light in all coordinate systems. Say we have some event P . Fig. 1.1 shows a light cone centered at event P . The vertical axis is time and the horizontal axis is a spatial dimension. The lines at $\pm 45^\circ$ represent the paths that light take both into the future and from the past. In the “Past” light cone, everything contained within the light cone is able to influence event P in a causal way (i.e. things traveling at or slower than the speed of light). Everything within the “Future” light cone can be influenced by event P . Events within these light cones are said to be causally connected to P . However, everything outside the light cone is unable to affect or be influenced by event P , thus we call those events causally disconnected.

In Newtonian gravity, we have a simple notion of an invariant distance between two points, e.g.

$$(\Delta s)^2 = (\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2. \quad (1.11)$$

This measure of distance is independent of what coordinate system we use for our spatial coordinates and works for Newtonian gravity because all time coordinates of all observers are the same. There is a universal time in Newtonian gravity. However, in SR we no longer have this universal time which everybody can agree on. Therefore, the measure of “distance” that we use in SR must account for our time coordinate somehow because of the coupling between space and time that the finite speed of light demands. For now, we just present this distance measure, called

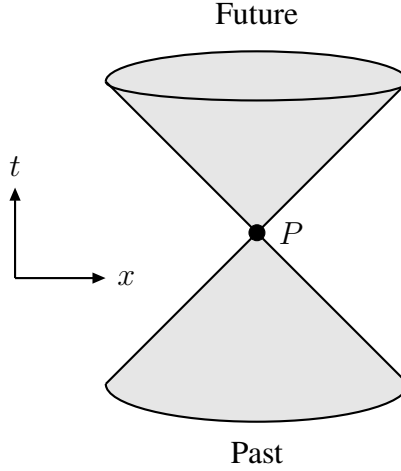


Figure 1.1: A light cone centered at the event P . The cone edges represent rays propagating at the speed of light c .

the spacetime interval, as

$$(\Delta s)^2 = -(\Delta t)^2 + (\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2. \quad (1.12)$$

First, notice that the units of this don't match. We are adding something like (seconds)² to something like (meters)². But one can notice that if we multiply the Δt by the speed of light c , we'll get the proper units. Very quickly in relativity, it becomes annoying to always have to write and keep track of this factor of c . Therefore, almost everybody in the field simply sets $c = 1$ with the assumption that if we ever need to calculate actual quantities, we'll add it back in at the end.

Second, we will now use this spacetime interval to motivate constructing something called a spacetime “metric”. Now to fully understand the significance of what a metric is and what it represents requires taking courses on differential geometry. However, for our purposes, a metric is basically a 4×4 matrix that represents how coordinates combine together to form the spacetime interval. Therefore, for the interval in Eq. 1.12, we can define its spacetime metric, named the Minkowski metric, as

$$\eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (1.13)$$

It is diagonal because there are no cross-terms in Eq. 1.12 (such as $\Delta x \Delta y$). We can then rewrite Eq. 1.12 using Einstein summation notation (where all repeated indices

are implicitly summed)

$$(\Delta s)^2 = \eta_{\mu\nu} \Delta x^\mu \Delta x^\nu \quad (1.14)$$

where

$$x^\mu = \begin{pmatrix} t \\ x \\ y \\ z \end{pmatrix}. \quad (1.15)$$

With a spacetime metric, we are able to do so many things. We can raise and lower indices of different quantities (i.e. transform between vectors and dual vectors/one-forms, $x_\nu = \eta_{\mu\nu} x^\mu$). We can turn the spacetime interval from Δs into infinitesimals and integrate it to find the length of a spacetime curve. But one of the most powerful things we can do is define transformations between coordinates that leave spacetime intervals unchanged (similar to Newtonian gravity). Let's define a transformation from an un-primed coordinate system to a primed one with

$$x^{\mu'} = \Lambda^{\mu'}_{\mu} x^\mu. \quad (1.16)$$

To show that this leaves the spacetime interval unchanged, let's plug it in

$$\begin{aligned} (\Delta s)^2 &= \eta_{\mu\nu} \Delta x^\mu \Delta x^\nu \\ &= \eta_{\mu'\nu'} \Delta x^{\mu'} \Delta x^{\nu'} \\ &= \eta_{\mu'\nu'} \Lambda^{\mu'}_{\mu} \Delta x^\mu \Lambda^{\nu'}_{\nu} \Delta x^\nu \\ &= \eta_{\mu'\nu'} \Lambda^{\mu'}_{\mu} \Lambda^{\nu'}_{\nu} \Delta x^\mu \Delta x^\nu. \end{aligned} \quad (1.17)$$

Therefore, Λ is a transformation between coordinates only if

$$\eta_{\mu\nu} = \eta_{\mu'\nu'} \Lambda^{\mu'}_{\mu} \Lambda^{\nu'}_{\nu}, \quad (1.18)$$

Or, said another way, Λ doesn't transform the components of the metric. If Λ satisfies this, then we call it a Lorentz transformation.

1.3 General Relativity

Now in going from Newtonian gravity to SR, we didn't actually add in any gravity to SR. We only tackled the issues of simultaneity and the speed of light. Now we are ready to combine SR with gravity to produce Einstein's theory of general relativity (GR). Probably the biggest conceptual difference between GR and Newtonian gravity is that Newtonian gravity is explained as a force exerted between two masses. And

any object that has a force being applied to it will accelerate in the direction of that force. Therefore, a change of motion in Newtonian gravity arises from forces.

In GR, however, motion does not arise from the force of gravity. Instead, in GR we say there is some “background” spacetime, or some fabric of the universe that matter exists both on and in at the same time. In the Newtonian and SR picture, this background is “flat” meaning that it doesn’t affect the matter that lives in it. However, in GR this background doesn’t have to be flat anymore. It can be distorted, or curved; sort of how a trampoline with a heavy object in the middle is curved. Motion then arises not because of any force, but simply due to the curvature of the background spacetime. The motion of a small mass is dictated by how much spacetime is curved. In the trampoline example, a small mass starting with a constant velocity on a flat spacetime (flat trampoline) will not deviate from its original path and will continue with a constant velocity. That same mass moving on a curved spacetime (trampoline with a heavy object in the middle) will naturally move towards the heavy object with increasing velocity without any outside influence. It’s the nature of the curved spacetime which drives the motion of matter. And the beauty of Einstein’s theory is that it is matter itself which drives the curvature of spacetime.

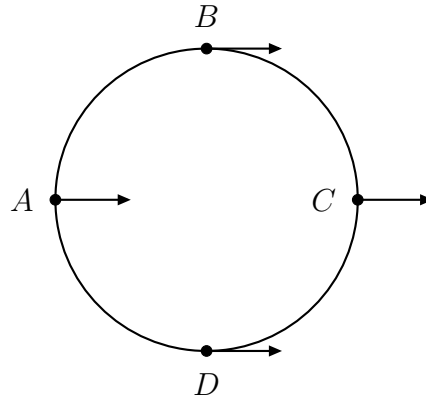


Figure 1.2: Diagram for parallel transporting a vector along two different paths of a flat circle: $A \rightarrow B \rightarrow C$ and $A \rightarrow D \rightarrow C$.

Therefore in GR, in order to describe gravity, we actually must describe the geometry of spacetime as this is what we experience as gravity. The first thing we should probably understand is how motion in a curved spacetime is different from a flat spacetime. Let’s demonstrate this by moving some vectors along some curves. In Fig. 1.2, we take a vector pointing to the right at point A and move it to point C along two different paths; $A \rightarrow B \rightarrow C$ and $A \rightarrow D \rightarrow C$. We move the vectors in such a way that doesn’t rotate or scale them. Therefore, somewhat intuitively, we

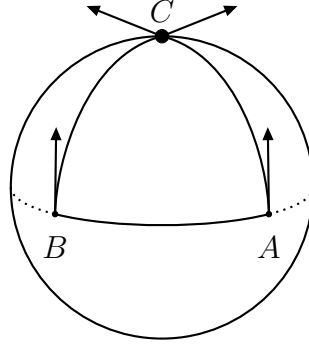


Figure 1.3: Diagram for parallel transporting a vector along two different paths of a sphere: $A \rightarrow B \rightarrow C$ and $A \rightarrow C$.

expect both vectors at C to be identical because the vectors weren't changed in any way except for their positions.

Now we do the same thing in Fig. 1.3, except this time on the surface of a sphere. We start with a vector at A that lies on the equatorial line of the sphere and move it to point C at the pole along two different paths; $A \rightarrow C$ and $A \rightarrow B \rightarrow C$. Along the first path, the vector will be pointing tangent to the pole towards the left of the image. Along the second path, the arrow is first moved along the equatorial line to B , and then moved to the pole pointing towards the right. Unlike in Fig. 1.2, the final vectors are not the same. This demonstrates the unintuitive nature of a curved space compared to a flat one, and why we will need some more machinery to explain GR.

The reason we are concerned with how vectors change as we move through our space is because any coordinate system that we create is made up of basis vectors. In a flat space, no matter what closed-loop path we choose, our basis vectors will always be the same at the same point. As shown in Fig. 1.3, this is not true for a curved space. Therefore, we must be careful about how we move vectors around our spacetime. To describe the curvature, similar to SR, GR will also have a metric associated with it. We will denote this metric as $g_{\mu\nu}$ because $\eta_{\mu\nu}$ is reserved for Minkowski spacetime. Our new metric $g_{\mu\nu}$ will encode everything we need to know about the gravity and curvature of our spacetime.

We start by expanding our notion of what a derivative is in our curved spacetime. Instead of the typical partial derivative $\partial/\partial x$, which works in a flat spacetime, a derivative which works in a curved spacetime will be independent of our coordinates and will reduce back to the partial derivative when the spacetime is flat. Since the

partial derivative is already a good place to start, we can say that our new derivative is the partial derivative plus some correction that accounts for the coordinates. We define the covariant derivative operator ∇_μ acting on the components of a vector V^λ as

$$\nabla_\mu V^\lambda = \partial_\mu V^\lambda + \Gamma_{\mu\nu}^\lambda V^\nu \quad (1.19)$$

where we have again made use of the Einstein summation notation of implicit sums over repeated indices. Here we call $\Gamma_{\mu\nu}^\lambda$ the connection coefficients as they connect basis vectors from different points in our spacetime. These connection coefficients are a general property of differential geometry and depend on the metric chosen, but in GR we call them the Christoffel symbols.

Before we define the Christoffel symbols in terms of the metric $g_{\mu\nu}$, we will impose two conditions that turn out to be necessary for GR. First is that our spacetime (and by extension the metric and the connection coefficients) are torsion free, meaning

$$\Gamma_{\mu\nu}^\lambda = \Gamma_{\nu\mu}^\lambda. \quad (1.20)$$

The second condition is that our connection is “metric compatible”, meaning

$$\nabla_\rho g_{\mu\nu} = 0. \quad (1.21)$$

This second condition is important for raising and lowering indices of derivative quantities. With these two conditions, it turns out that we can write the Christoffel symbols as sums of partial derivatives of the metric

$$\Gamma_{\mu\nu}^\lambda = \frac{1}{2} g^{\lambda\rho} (\partial_\mu g_{\nu\rho} + \partial_\nu g_{\rho\mu} - \partial_\rho g_{\mu\nu}). \quad (1.22)$$

Next we will define how to transport a vector in the most “constant” way possible along a path in a curved spacetime. This is useful because we still want to be able to move vectors in our spacetime without them being rotated or changed in some way. We define parallel transport of a vector V^σ along a path $x(\lambda)$ in our spacetime

$$\frac{dx^\mu}{d\lambda} \nabla_\mu V^\sigma = \frac{dx^\mu}{d\lambda} \frac{\partial V^\sigma}{\partial x^\mu} + \Gamma_{\mu\nu}^\sigma \frac{dx^\mu}{d\lambda} V^\nu = 0 \quad (1.23)$$

If the vector that we apply parallel transport to is the tangent vector itself, $V^\mu = dx^\mu/d\lambda$, then we get the equation

$$\frac{d^2 x^\mu}{d\lambda^2} + \Gamma_{\rho\sigma}^\mu \frac{dx^\rho}{d\lambda} \frac{dx^\sigma}{d\lambda} = 0, \quad (1.24)$$

which is called the geodesic equation. The geodesic equation will be our notion of a “straight line” in a curved spacetime, i.e. a path without any external forces or influence. If you squint hard enough, you may be able to see that this is the GR extension of zero acceleration, $\mathbf{a} = d^2\mathbf{x}/dt^2 = 0$, in Newtonian gravity.

Now that we are able to compute how to move vectors around our spacetime, we can finally move on to computing the curvature of our spacetime, which is where gravity in GR derives from. To do this, we start with a vector V^ρ . We already know if we move this vector along two different paths to the same destination, the resulting vectors will be different. This difference is precisely what we are going to use to define curvature. Imagine we first transport the vector in some direction ∇_ν and then in another direction ∇_μ . This gives us a final vector of $\nabla_\mu \nabla_\nu V^\rho$. Now, we swap the order to get $\nabla_\nu \nabla_\mu V^\rho$. Let’s take the difference and expand the covariant derivatives

$$\begin{aligned}\nabla_\mu \nabla_\nu V^\rho - \nabla_\nu \nabla_\mu V^\rho &= \partial_\mu (\nabla_\nu V^\rho) - \Gamma_{\mu\nu}^\lambda \nabla_\lambda V^\rho + \Gamma_{\mu\sigma}^\rho \nabla_\nu V^\sigma - (\mu \leftrightarrow \nu) \\ &= \partial_\mu \partial_\nu V^\rho + \partial_\mu (\Gamma_{\nu\sigma}^\rho) V^\sigma + \Gamma_{\nu\sigma}^\rho \partial_\mu V^\sigma - \Gamma_{\mu\nu}^\lambda \partial_\lambda V^\rho \\ &\quad - \Gamma_{\mu\nu}^\lambda \Gamma_{\lambda\sigma}^\rho V^\sigma + \Gamma_{\mu\sigma}^\rho \partial_\nu V^\sigma + \Gamma_{\mu\sigma}^\rho \Gamma_{\nu\lambda}^\sigma V^\lambda - (\mu \leftrightarrow \nu) \\ &= (\partial_\mu \Gamma_{\nu\sigma}^\rho - \partial_\nu \Gamma_{\mu\sigma}^\rho + \Gamma_{\mu\lambda}^\rho \Gamma_{\nu\sigma}^\lambda - \Gamma_{\nu\lambda}^\rho \Gamma_{\mu\sigma}^\lambda) V^\sigma\end{aligned}\quad (1.25)$$

In the last line we simplified quite a bit, especially using the torsion-free condition, but now we have the difference of moving a vector along two different paths. Let’s rewrite this in a more compact manner

$$\nabla_\mu \nabla_\nu V^\rho - \nabla_\nu \nabla_\mu V^\rho = R^\rho_{\sigma\mu\nu} V^\sigma \quad (1.26)$$

where we call $R^\rho_{\sigma\mu\nu}$ the Riemann curvature tensor. To write it out more explicitly,

$$R^\rho_{\sigma\mu\nu} = \partial_\mu \Gamma_{\nu\sigma}^\rho - \partial_\nu \Gamma_{\mu\sigma}^\rho + \Gamma_{\mu\lambda}^\rho \Gamma_{\nu\sigma}^\lambda - \Gamma_{\nu\lambda}^\rho \Gamma_{\mu\sigma}^\lambda. \quad (1.27)$$

The Riemann curvature tensor is made up of first and second derivatives of the metric because the Christoffel symbols (Eq. 1.22) have first derivatives of the metric in them. The Riemann tensor encodes everything we need to know about the curvature of our spacetime, and it is only computed from derivatives of our metric $g_{\mu\nu}$. There are a number of symmetries of the Riemann tensor which we won’t go over, but they reduce the number of independent components from $4^4 = 256$ to just 20.

A useful contraction of the Riemann tensor is called the Ricci tensor

$$R_{\mu\nu} = R^\lambda_{\mu\lambda\nu}. \quad (1.28)$$

The Ricci tensor has the property of being symmetric, $R_{\mu\nu} = R_{\nu\mu}$, and only has 10 independent components. Again, we can contract the Ricci tensor to get the Ricci scalar

$$R = R^\mu{}_\mu. \quad (1.29)$$

Finally, we are able to combine our work to form the Einstein tensor

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu}, \quad (1.30)$$

which has the nice property of $\nabla^\mu G_{\mu\nu} = 0$.

Now that we have the Einstein tensor, let us take a step back. In the Einstein tensor, we have sums of the metric $g_{\mu\nu}$, its first derivative squared $(\partial_\rho g_{\mu\nu})^2$, and second derivatives $\partial_\lambda \partial_\rho g_{\mu\nu}$. We know that all these derivatives somehow represent the curvature of our spacetime which in turn gives us gravity. Let's relate this back to Newtonian gravity to see if we can gain a little intuition. Recall Poisson's equation for a gravitational potential Φ (rewritten here for ease)

$$\nabla^2 \Phi = 4\pi G\rho. \quad (1.31)$$

Φ encodes the information of our gravitational field. On the left, we have two derivatives of this gravitational potential and on the right we have the density of matter ρ . We also know that Newtonian gravity is a good approximation for GR in the case of weak gravity like we have here on Earth. We happen to have just constructed a quantity that contains second derivatives of the metric $g_{\mu\nu}$ which is our GR analog of the gravitational potential Φ .

Therefore, we can postulate, just as Einstein did, that the equations of GR have the following form:

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi GT_{\mu\nu}. \quad (1.32)$$

These are the infamous Einstein field equations. They are a set of 10 coupled non-linear second order partial differential equations (PDEs) for the spacetime metric. On the right, we have introduced a quantity $T_{\mu\nu}$ which is called the stress energy tensor. This tensor contains all the information about the matter and energy distribution in our spacetime. Because we have that $\nabla^\mu G_{\mu\nu} = 0$, we also have that $\nabla^\mu T_{\mu\nu} = 0$. Based on how we construct $T_{\mu\nu}$, this actually works out to be the conditions for conservation of energy and momentum, all wrapped up into one tensor equation. Also, it should be noted that G is Newton's gravitational constant, and not the trace of $G_{\mu\nu}$. We often set $G = 1$ as well to avoid confusion and make the equations easier to work with.

There is considerable physics to be studied when we include the stress energy tensor in the Einstein field equations such as neutron stars, white dwarf stars, cosmology, active galactic nuclei, black hole accretion disks, and supernovae to name a few. However, for this thesis we will be focusing on black holes. And even though many black holes form due to the collapse and/or accretion of matter, once a black hole is formed there need not be any matter outside the black hole to describe a black hole spacetime. Therefore, for the majority of this thesis, we will set $T_{\mu\nu} = 0$. This reduces the Einstein field equations significantly to

$$R_{\mu\nu} = 0. \quad (1.33)$$

As an example, let us look at the simplest black hole spacetime that exists; the Schwarzschild spacetime. This is a static spherically symmetric spacetime. The metric in spherical coordinates for the Schwarzschild solution is

$$ds^2 = - \left(1 - \frac{2M}{r}\right) dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 (d\theta^2 + \sin^2 \theta d\phi^2), \quad (1.34)$$

or written in tensor form

$$g_{\mu\nu} = \begin{pmatrix} -\left(1 - \frac{2M}{r}\right) & 0 & 0 & 0 \\ 0 & \left(1 - \frac{2M}{r}\right)^{-1} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2 \theta \end{pmatrix}. \quad (1.35)$$

Here, M is the mass of our black hole. For comparison, the Minkowski metric of flat spacetime in spherical coordinates is

$$ds^2 = -dt^2 + dr^2 + r^2 d\Omega^2. \quad (1.36)$$

where $d\Omega^2 = (d\theta^2 + \sin^2 \theta d\phi^2)$. An interesting property of the Schwarzschild spacetime is that it is actually the unique solution for any spherically symmetric vacuum spacetime. The static nature of Schwarzschild arises as a byproduct of the spherical symmetry condition.

Looking at the Schwarzschild solution, something interesting happens at $r = 2M$. The g_{rr} component blows up to infinity. When the metric components go to infinity, we call this a singularity. This radius is so important, that it's named the Schwarzschild radius,

$$R_S = 2M. \quad (1.37)$$

This radius defines the event horizon of the black hole. The event horizon is an invisible surface within which nothing (including light) can escape out to $r \geq 2M$.

Also, if $r = 0$, the g_{tt} component of the metric goes to infinity; another singularity. However, this singularity is different than R_S . We discussed in § 1.2 the importance of coordinate freedom and being able to switch coordinates. Let us make the following coordinate transformations

$$r^* = r + 2M \ln \left(\frac{r}{2M} - 1 \right), \quad (1.38)$$

$$v = t + r^*, \quad (1.39)$$

$$u = t - r^*. \quad (1.40)$$

With these transformations, our metric becomes

$$ds^2 = - \left(1 - \frac{2M}{r} \right) dv^2 + (dvdr + drdv) + r^2 d\Omega^2. \quad (1.41)$$

Now, we only have one singularity, specifically when $r = 0$. Therefore, that means the $r = 2M$ singularity was what we call a “coordinate” singularity; i.e. not a “physical” singularity like $r = 0$. This is why we say that the event horizon is an invisible boundary because observers falling into a black hole will be uninterrupted at $r = 2M$ and continue to fall. However, at $r = 0$, our classical theory of GR breaks down. In order to describe what happens at (or after) $r = 0$, we would need a valid theory of quantum gravity, which we still do not yet have.

There is plenty more analysis one can do with the Schwarzschild solution to gain some intuition on how we should think of a curved spacetime. This includes computing the components of the Riemann tensor, Ricci tensor, and Christoffel symbols, computing the geodesics and stable orbits (Eq. 1.24), computing the Killing vectors, performing more coordinate transformations and expanding the Schwarzschild solution to negative radius, analyzing Kruskal diagrams, and more. All of this should give better intuition about the dynamics in a curved spacetime. There is also an extension of the Schwarzschild solution to axial symmetry which is called the Kerr solution, again exploring all the same things for this new spacetime. For our purposes though, explaining R_S and the $r = 0$ physical singularity shall suffice.

1.4 Numerical Relativity

We now have a set of PDEs that we can solve to give us the metric at every point in our spacetime. Our job now is to put these equations onto the computer so we can solve them numerically. This is necessary because the Schwarzschild and Kerr

solutions are the only analytic black hole solutions available, which means we will need numerical relativity (NR) to generate solutions of two black holes merging.

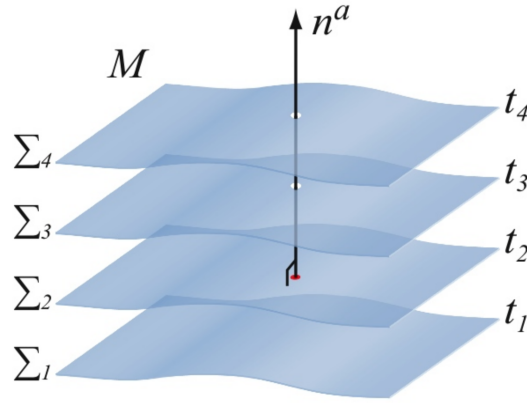


Figure 1.4: A foliation of spacetime into spatial hypersurfaces Σ_t , each labeled by a coordinate time t . The vector n^a is normal to each spatial hypersurface. This image was taken from [1].

To put Einstein’s equations onto the computer, we first need to split up the equations to somewhat disentangle the space and time dimensions from our spacetime. This procedure is called the “3+1” decomposition of spacetime. We do this by creating what is called a spacetime foliation. Fig. 1.4 depicts how we will foliate our spacetime. We start by splitting the spacetime up into spatial hypersurfaces Σ for every time t of our spacetime. Each Σ is a snapshot of full 3D space at every time (even though in the image they are represented as 2D slices, and time is the vertical axis). Then we can construct a vector n^a which is normal to our Σ hypersurfaces and points into the future. Note that for this section, we will switch from using greek letters (e.g. μ, ν, ρ) to latin letters (e.g. a, b, c) for spacetime symbols. This is purely a convention the NR community has adopted.

We must be careful with our time coordinate. Recall from § 1.2 that SR has no global time that applies to all spacetime points. This is still true in GR, so we must take this into account. So instead of our time t being a single number, we will make it a function. We can use t to describe the normal vector n^a . We would like n_a to be normalized and timelike ($n^a n_a = -1$) so we define a normalization factor α , called the lapse, to achieve this. Then, n^a becomes

$$n^a = -\frac{1}{\alpha} \nabla^a t. \quad (1.42)$$

Here, the lapse measures how much proper time has elapsed between spatial hypersurfaces.

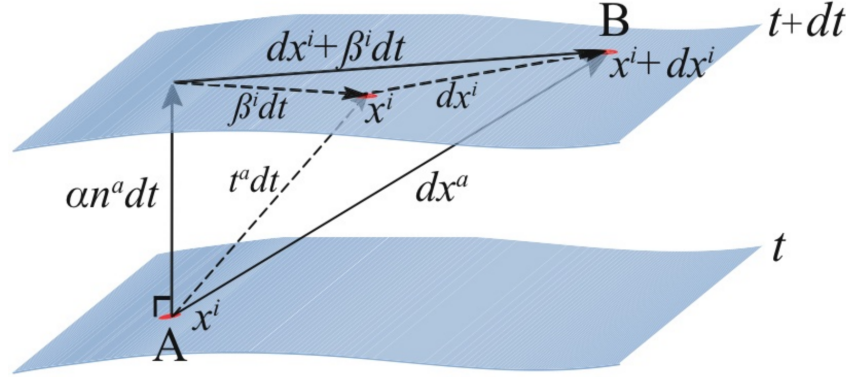


Figure 1.5: Illustration of the lapse α and shift vector β^i in the 3+1 decomposition of spacetime. The lapse α measures the proper time between adjacent hypersurfaces, while the shift β^i encodes how coordinates move between hypersurfaces. This image was taken from [1].

Even though we now have how much time elapses between Σ , there's still more to our coordinates. Because of the coordinate freedom of GR, coordinates don't have to stay stationary between different Σ . In fact, we will show soon that we actually evolve the coordinates, implying that they will be different as time goes on. Our timelike normal vector n^a isn't enough to describe this movement. Therefore, we introduce a spatial vector $\beta^a = (0, \beta^i)$ called the shift which will tell us how much our coordinates have moved (or shifted) from the normal vector n^a . We can therefore construct a new vector t^a that will tell us how the coordinate that a grid point represents moves between spatial hypersurfaces

$$t^a = \alpha n^a + \beta^a. \quad (1.43)$$

Fig. 1.5 is a great diagram that shows the breakdown of t^a into the timelike and spacelike components. It shows how those components map a spacetime coordinate at grid point A of one hypersurface to grid point B of the next hypersurface. Notice how αn^a gets us from the first hypersurface to the next, and β^i moves us within the next hypersurface. The lapse and shift together are four degrees of freedom, which is the exact number of coordinate degrees of freedom that GR lets us specify.

Now that we have the lapse and shift, we can move on to specifying our spacetime metric g_{ab} in terms of the lapse and shift. It will be very convenient to define a spatial metric within our spatial hypersurface Σ which we can use for quantities that live in Σ . We can combine our timelike normal vector with the spacetime metric to get

$$\gamma_{ab} = g_{ab} + n_a n_b. \quad (1.44)$$

There's a plus sign because n^a is defined with a minus sign. This means that γ_{ab} is purely spatial.

With this, we can represent our spacetime metric in terms of $\gamma_{ij}, \alpha, \beta^i$. If we choose $t^a = (1, 0, 0, 0)$, we can then compute n^a from Eq. 1.43 and solve Eq. 1.44 for g_{ab} as

$$g_{ab} = \begin{pmatrix} -\alpha^2 + \beta_l \beta^l & \beta_j \\ \beta_j & \gamma_{ij} \end{pmatrix}. \quad (1.45)$$

Or, expressed more commonly as a line element,

$$ds^2 = g_{ab} dx^a dx^b = -\alpha^2 dt^2 + \gamma_{ij} (dx^i + \beta^i dt) (dx^j + \beta^j dt). \quad (1.46)$$

At this point, we are still only at the beginning of manipulating our equations. We would like to end up with equations that contain time derivatives ∂_t on the left hand side of the equation. These types of equations are ones that we can integrate in time to solve for our spacetime metric. We now branch off into explaining the two most popular formalisms for writing down evolution equations.

1.4.1 BSSN Formulation

The first formalism we will discuss is the Baumgarte-Shapiro-Shibata-Nakamura (BSSN) formalism [2, 3]. First, we finish off defining the 3+1, or Arnowitt-Deser-Misner [4] (ADM), formalism as it's necessary for the BSSN formalism. This section is heavily based on derivations from [1]. While we won't derive them here, the BSSN formulation is closely related to the Z4 family of formulations [5–8].

We start by defining the extrinsic curvature K_{ab} . The extrinsic curvature is different than the Riemann curvature, or intrinsic curvature. The Riemann curvature tells us about how a spacetime itself is curved, and only relies on quantities defined in the spacetime. Said another way, intrinsic curvature does not rely on higher dimensions to define the curvature. However, our spatial slice Σ is a 3D space embedded in a higher dimensional (4D) space. Therefore, the extrinsic curvature will tell us how Σ is curved within the broader 4D spacetime. We define the extrinsic curvature by taking gradients of our normal vector n^a , and then projecting them onto Σ

$$K_{ab} = -\gamma_a^c \gamma_b^d \nabla_c n_d. \quad (1.47)$$

Using our definitions for γ_{ab} , we can rewrite this only in terms of n^a as

$$K_{ab} = -\nabla_a n_b - n_a n^c \nabla_c n_b. \quad (1.48)$$

We also introduce a new derivative operator, the Lie derivative $\mathcal{L}_{\mathbf{X}}$ as

$$\mathcal{L}_{\mathbf{X}}T^a{}_b = X^c\partial_c T^a{}_b - T^c{}_b\partial_c X^a + T^a{}_c\partial_b X^c. \quad (1.49)$$

With the Lie derivative, we can write the extrinsic curvature as

$$K_{ab} = -\frac{1}{2}\mathcal{L}_{\mathbf{n}}\gamma_{ab}. \quad (1.50)$$

We can also define the trace of the extrinsic curvature as

$$K = \gamma^{ab}K_{ab} = -\mathcal{L}_{\mathbf{n}}\ln\gamma^{1/2}. \quad (1.51)$$

To get time derivatives of the extrinsic curvature and our spatial metric, we will take the Lie derivative of each along our vector t^a . For the extrinsic curvature this will be (with a lot of simplification)

$$\begin{aligned} \mathcal{L}_{\mathbf{t}}K_{ab} &= \mathcal{L}_{\alpha\mathbf{n}+\beta}K_{ab} \\ &= \alpha\mathcal{L}_{\mathbf{n}}K_{ab} + \mathcal{L}_{\beta}K_{ab} \\ &= -D_a D_b \alpha + \alpha({}^{(3)}R_{ab} - 2K_{ac}K^c{}_b + K K_{ab}) + \mathcal{L}_{\beta}K_{ab} \end{aligned} \quad (1.52)$$

with ${}^{(3)}R_{ab}$ being the 3-dimensional Ricci tensor and $D_a f = \gamma_a{}^b\nabla_b f$. For the spatial metric, this becomes

$$\begin{aligned} \mathcal{L}_{\mathbf{t}}\gamma_{ab} &= \mathcal{L}_{\alpha\mathbf{n}+\beta}\gamma_{ab} \\ &= \alpha\mathcal{L}_{\mathbf{n}}\gamma_{ab} + \mathcal{L}_{\beta}\gamma_{ab} \\ &= -2\alpha K_{ab} + \mathcal{L}_{\beta}\gamma_{ab}. \end{aligned} \quad (1.53)$$

Because of our definition of $t^a = (1, 0, 0, 0)$ and the fact that β is spatial, the following derivatives simplify to

$$\mathcal{L}_{\mathbf{t}} = \partial_t \quad (1.54)$$

$$\mathcal{L}_{\beta}K_{ij} = \beta^k D_k K_{ij} + K_{ik} D_j \beta^k + K_{kj} D_i \beta^k \quad (1.55)$$

$$\mathcal{L}_{\beta}\gamma_{ij} = D_i \beta_j + D_j \beta_i. \quad (1.56)$$

We could also take the trace of both the extrinsic curvature and spatial metric to get equations for the evolution of their traces

$$\partial_t \ln \gamma^{1/2} = -\alpha K + D_i \beta^i \quad (1.57)$$

$$\partial_t K = -D^2 \alpha + \alpha K_{ij} K^{ij} + \beta^i D_i K. \quad (1.58)$$

There are actually extra terms in these equations for the stress-energy tensor following a similar procedure, but we exclude those here because we are only concerned with the vacuum equations.

It turns out that the ADM equations do not lead to stable numerical evolutions for reasons that are well understood mathematically [1, 9]. Accordingly, we now move on to the BSSN equations [2, 3]. We start by rewriting the spatial metric as

$$\gamma_{ij} = e^{4\phi} \bar{\gamma}_{ij} \quad (1.59)$$

where e^ϕ is a scalar function called the conformal factor, and $\bar{\gamma}_{ij}$ is called the conformal metric. This procedure of rewriting a quantity as a different quantity times a scalar function is a conformal transformation. The fourth power of e^ϕ is arbitrary, but useful for later calculation. The determinant of $\bar{\gamma}_{ij}$ is required to be $\bar{\gamma} = 1$. We then define the scalar function ϕ to be

$$\phi = \frac{1}{12} \ln(\gamma). \quad (1.60)$$

We also rewrite the extrinsic curvature as

$$K_{ij} = A_{ij} + \frac{1}{3} K \gamma_{ij} \quad (1.61)$$

where A_{ij} is the trace-free part of the extrinsic curvature. We also conformally transform this trace-free part as

$$A_{ij} = e^{4\phi} \bar{A}_{ij}. \quad (1.62)$$

Eq. 1.56 for the time derivative of the spatial metric will now split into two equations, one for ϕ and one for $\bar{\gamma}_{ij}$ because of the conformal transformation. Those new equations are

$$\partial_t \phi = -\frac{1}{6} \alpha K + \beta^i \partial_i \phi + \frac{1}{6} \partial_i \beta^i \quad (1.63)$$

$$\partial_t \bar{\gamma}_{ij} = -2\alpha \bar{A}_{ij} + \beta^k \partial_k \bar{\gamma}_{ij} + \bar{\gamma}_{ik} \partial_j \beta^k + \bar{\gamma}_{kj} \partial_i \beta^k - \frac{2}{3} \bar{\gamma}_{ij} \partial_k \beta^k \quad (1.64)$$

In a similar fashion, Eq. 1.55 for the time derivative of the extrinsic curvature will also split into two equations, one for K and one for $\bar{A}_{ij}$ because of the conformal transformation. The new equations are

$$\partial_t K = -\gamma^{ij} D_j D_i \alpha + \alpha \left(\bar{A}_{ij} \bar{A}^{ij} + \frac{1}{3} K^2 \right) + \beta^i \partial_i K, \quad (1.65)$$

$$\begin{aligned} \partial_t \bar{A}_{ij} = & e^{-4\phi} \left(-(D_i D_j \alpha)^{TF} + \alpha^{(3)} R_{ij}^{TF} \right) + \alpha (K \bar{A}_{ij} - 2 \bar{A}_{il} \bar{A}_j^l) \\ & + \beta^k \partial_k \bar{A}_{ij} + \bar{A}_{ik} \partial_j \beta^k + \bar{A}_{kj} \partial_i \beta^k - \frac{2}{3} \bar{A}_{ij} \partial_k \beta^k, \end{aligned} \quad (1.66)$$

where TF means the trace-free part of the tensor, e.g. ${}^{(3)}R_{ij}^{TF} = {}^{(3)}R_{ij} - {}^{(3)}R/3\gamma_{ij}$.

Next, we remember that with a conformal metric $\bar{\gamma}_{ij}$ there exists a conformal connection ${}^{(3)}\bar{\Gamma}_{jk}^i$ since connections are tied to metrics. We can contract the conformal connection to get

$${}^{(3)}\bar{\Gamma}^i \equiv \bar{\gamma}^{jk} {}^{(3)}\bar{\Gamma}_{jk}^i = -\partial_j \bar{\gamma}^{ij}. \quad (1.67)$$

In the BSSN formalism, we promote ${}^{(3)}\bar{\Gamma}^i$ from a computed quantity to an independent variable that is evolved along with all the others. If we take the time derivative of the previous equation, substitute in Eq. 1.64, and simplify a lot to get rid of the $\partial_j \bar{A}^{ij}$ term, we wind up with

$$\begin{aligned} \partial_t {}^{(3)}\bar{\Gamma}^i = & -2\bar{A}^{ij}\partial_j\alpha + 2\alpha\left({}^{(3)}\bar{\Gamma}_{jk}^i\bar{A}^{kj} - \frac{2}{3}\bar{\gamma}^{ij}\partial_j K + 6\bar{A}^{ij}\partial_j\phi\right) \\ & + \beta^j\partial_j {}^{(3)}\bar{\Gamma}^i - {}^{(3)}\bar{\Gamma}^j\partial_j\beta^i + \frac{2}{3}{}^{(3)}\bar{\Gamma}^i\partial_j\beta^j + \frac{1}{3}\bar{\gamma}^{li}\partial_l\partial_j\beta^j + \bar{\gamma}^{lj}\partial_j\partial_l\beta^i. \end{aligned} \quad (1.68)$$

Eqs. 1.63, 1.64, 1.65, 1.66, and 1.68 make up the evolution equations of the BSSN formalism. The last piece we'll need are evolution equations for the gauge functions α and β^i . Coming up with evolution equations for gauge quantities is more of an art than a science since, in theory, no matter what gauge we choose, we should still get the same physical system. However, solving equations numerically usually imposes more practical constraints.

Several gauge conditions are possible with BSSN but are not commonly used. One is the geodesic slicing condition, $\alpha = 1$ and $\beta^i = 0$. Another is maximal slicing where $K = 0$ turns into an elliptic equation for the lapse, $D^2\alpha = \alpha K_{ij}K^{ij}$. Harmonic coordinates (which will be discussed more in § 1.4.2) set $\Gamma^a = 0$.

Several of the mentioned conditions for the lapse can be described by

$$(\partial_t - \beta^j\partial_j)\alpha = -\alpha^2 f(\alpha)K \quad (1.69)$$

where $f(\alpha)$ is a generic function. Choosing $f(\alpha) = 2/\alpha$, we can integrate Eq. 1.69 to get

$$\alpha = 1 + \ln \gamma. \quad (1.70)$$

This is appropriately called the $1 + \log$ slicing condition and is very popular paired with the BSSN formalism.

A choice for the shift is to set $\partial_t {}^{(3)}\bar{\Gamma}^i = 0$. This is named “gamma freezing”. Plugging this into Eq. 1.68 will yield a very complicated equation to solve for

β^i . We can generalize and approximate this to something called a “gamma driver” equation

$$(\partial_t - \beta^j \partial_j) \beta^i = k(\partial_t {}^{(3)}\bar{\Gamma}^i + \frac{1}{2} {}^{(3)}\bar{\Gamma}^i), \quad (1.71)$$

where k is a positive constant. Since plugging in Eq. 1.68 still gives us a parabolic equation, we can make one more modification to get the hyperbolic gamma driver condition

$$(\partial_t - \beta^j \partial_j) \beta^i = \frac{3}{4} B^i \quad (1.72)$$

$$(\partial_t - \beta^j \partial_j) B^i = \partial_t {}^{(3)}\bar{\Gamma}^i - \frac{1}{2} B^i. \quad (1.73)$$

The $1 + \log$ slicing for the lapse and both gamma driver equations are the most popular gauge conditions to choose for the BSSN formalism. Small tweaks to each of these exist in the different codes that use the BSSN formalism. One such tweak is removing the advective term $\beta^j \partial_j$ from the left hand side of the equations. These two conditions combined are referred to as the “moving puncture” conditions since they follow where the singularities of each black hole would be.

1.4.2 Generalized Harmonic Formulation

The next (and final) formalism we will discuss is the generalized harmonic (GH) formulation [10]. This differs quite a bit from the BSSN formalism, but we will still borrow some things as we need them. Namely, we will still keep the standard 3+1 decomposition of the metric since having the lapse, shift, and spatial metric will be needed.

$$ds^2 = g_{ab} dx^a dx^b = -\alpha^2 dt^2 + \gamma_{ij} (dx^i + \beta^i dt) (dx^j + \beta^j dt). \quad (1.74)$$

Right away, we will start with the gauge as this formalism is based around using harmonic coordinates. Given coordinates x^a where each x^a is treated as a scalar function, we have the relation that

$$\begin{aligned} \square x^a &= g^{bc} \nabla_b \nabla_c x^a = g^{bc} \nabla_b (\partial_c x^a) \\ &= g^{bc} (\partial_b \partial_c x^a - \Gamma_{bc}^d \partial_d x^a) \\ &= -g^{bc} \Gamma_{bc}^a \\ &= -\Gamma^a \end{aligned} \quad (1.75)$$

$\square$ is the d’Alembertian operator and we have used the fact that since x^a is a scalar and are coordinates, $\nabla_c x^a = \partial_c x^a = \delta_c^a$. We then express the harmonic gauge

condition as

$$\square x^a = -\Gamma^a = 0. \quad (1.76)$$

The generalized harmonic gauge condition is then related to the harmonic gauge condition, but instead of setting $-\Gamma^a = 0$, we set it as

$$\square x^a = -\Gamma^a = H^a(x, g), \quad (1.77)$$

where H^a is some arbitrary function of the coordinates and spacetime metric (but not derivatives of the metric!) that we specify. These conditions on H^a are critical because they preserve the strong hyperbolicity of the Einstein equations, which allows for stable evolutions of spacetime. If these conditions were not met, any tiny instability or error in our evolutions would grow exponentially and spoil our solution. And since we are using computers to solve these problems, the finite truncation of floating point numbers will always introduce some tiny error so we can be guaranteed an unstable evolution.

Now in the BSSN formalism, we showed that the lapse and the shift contain our coordinate freedom. Then to impose a gauge condition, we had to specify the lapse and shift either analytically or by evolving them alongside the spacetime quantities. Here we show that specifying H^a is equivalent to specifying the lapse and shift with the following evolution equations[10]

$$\partial_t \alpha - \beta^j \partial_j \alpha = -\alpha(H_0 - \beta^i H_i + \alpha K) \quad (1.78)$$

$$\partial_t \beta^i - \beta^j \partial_j \beta^i = \alpha \gamma^{ij} [\alpha(H_j + \gamma^{kl} \Gamma_{jkl}) - \partial_j \alpha] \quad (1.79)$$

where K is the extrinsic curvature and $\gamma^{ij} \Gamma_{jkl} = \Gamma_{kl}^i$. Here we can see that even though it may not be easy to convert between the lapse and shift and H^a , specifying one or the other is equivalent.

In the GH formulation, we won't be evolving quantities like γ_{ij} , K_{ij} , or Γ^i . Instead we will be evolving the full spacetime metric g_{ab} itself. The specific form of the GH formulation that we will be presenting is the first-order symmetric hyperbolic GH formulation. In the typical formulation, we would time integrate the second time derivative ∂_t^2 of the metric. In the first-order formulation, we only evolve ∂_t . This is advantageous because there are a number of well-known standard integrators that work on first-order hyperbolic equations. However, this requires us to somehow reduce the order of the PDEs since Einstein's equations are second order.

To show the procedure, we first start with an example of reducing the order of a scalar wave function

$$\square\psi = \left(\frac{\partial^2}{\partial t^2} - \nabla^2\right)\psi = 0. \quad (1.80)$$

We choose this equation as an example because Einstein's equations themselves are essentially complicated wave equations due to the second-order derivatives in both space and time that appear from the Riemann tensor. To reduce the order of the scalar wave system, we introduce two new auxiliary variables

$$\Pi \equiv -\partial_t\psi, \quad (1.81)$$

$$\Phi_i \equiv \partial_i\psi. \quad (1.82)$$

Using these new variables, we can take time derivatives of each to get a system of equations equivalent to Eq. 1.80

$$\partial_t\psi = -\Pi \quad (1.83)$$

$$\partial_t\Pi = -\partial^i\Phi_i \quad (1.84)$$

$$\partial_t\Phi_i = -\partial_i\Pi \quad (1.85)$$

We have transformed a second-order equation for ψ into a system of first-order equations for ψ, Π, Φ_i . It can be useful to combine these evolved variables in a single vector $u^A = \{\psi, \Pi, \Phi_i\}$. We can then rewrite the evolution equations as

$$\partial_t u^A + M^j \partial_j u^A = S(u^A) \quad (1.86)$$

where M^j is a coefficient matrix and $S(u^A)$ is a source vector that does not contain any derivatives of u^A . In this form, if all the left-eigenvectors of M^j are linearly independent, then the system is said to be strongly hyperbolic. We will return to this shortly.

Notice that Eq. 1.81 for Π became an evolution equation, but Eq. 1.82 for Φ_i isn't an evolution equation because it doesn't have time derivatives. It is actually called a constraint equation because now we evolve ψ and Φ_i independently. We can define two constraints with Φ_i

$$\mathcal{C}_i = \partial_i\psi - \Phi_i = 0 \quad (1.87)$$

$$\mathcal{C}_{ij} = \partial_i\Phi_j - \partial_j\Phi_i = 0 \quad (1.88)$$

Therefore, we can actually use $\mathcal{C}_i$ and $\mathcal{C}_{ij}$ throughout the simulation to check if our evolution is consistent. Additionally, and more importantly, the version of the

evolution equations above will leave errors around in our solution. To mitigate this, we can actually alter the third equation to depend on $\mathcal{C}_i$ which will damp away any errors that arise. We do this by adding a multiple of $\mathcal{C}_i$ to the Φ_i equation. The modified equations are

$$\partial_t \psi = -\Pi \quad (1.89)$$

$$\partial_t \Pi = -\partial^i \Phi_i \quad (1.90)$$

$$\partial_t \Phi_i = -\partial_i \Pi + \gamma_2 (\partial_i \psi - \Phi_i) \quad (1.91)$$

where γ_2 is some analytic function specified for all space and time.

With these new equations we have that

$$M^j = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ -\gamma_2 & 1 & 0 \end{pmatrix}, \quad S(u^A) = \begin{pmatrix} -\Pi \\ 0 \\ -\gamma_2 \Phi_i \end{pmatrix}. \quad (1.92)$$

The left eigenvectors, or characteristic fields, of this system are

$$w^\psi = \psi \quad (1.93)$$

$$w_i^0 = (\delta_i^k - n^k n_i) \Phi_k \quad (1.94)$$

$$w^\pm = \Pi \pm n^k \Phi_k - \gamma_2 \psi \quad (1.95)$$

where n^k is a vector in whatever direction we are interested in analyzing. The eigenvalues, or characteristic speeds, associated with these eigenvectors are

$$\lambda_\psi = 0 \quad (1.96)$$

$$\lambda_0 = 0 \quad (1.97)$$

$$\lambda_\pm = \pm 1 \quad (1.98)$$

And while we won't discuss their use here, it is also useful to express our evolved variables u^A in terms of the characteristic fields:

$$\psi = w^\psi \quad (1.99)$$

$$\Pi = \frac{1}{2}(w^+ + w^-) + \gamma_2 w^\psi \quad (1.100)$$

$$\Phi_i = \frac{1}{2}(w^+ - w^-)n_i + w_i^0. \quad (1.101)$$

Lastly, we take a time derivative of our constraint $\mathcal{C}_i$ to see how it evolves in time.

$$\partial_t \mathcal{C}_i = \partial_t \partial_i \psi - \partial_t \Phi_i \quad (1.102)$$

$$= -\partial_i \Pi - (-\partial_i \Pi + \gamma_2 \mathcal{C}_i) \quad (1.103)$$

$$= -\gamma_2 \mathcal{C}_i \quad (1.104)$$

where we used $\partial_t \psi = -\Pi$ and Eq. 1.91. The solution to this differential equation is $\propto e^{-\gamma_2 t}$, so our constraints will be exponentially damped. A similar equation exists for $\mathcal{C}_{ij}$

$$\partial_t \mathcal{C}_{ij} = -\gamma_2 \mathcal{C}_{ij}. \quad (1.105)$$

With these characteristic fields, speeds, and constraints, one is also able to impose boundary conditions on the incoming characteristic fields rather than evolved variables themselves. However, here we skip that as we are more interested in the evolution equations themselves.

Now we can return back to the GH formulation. For gravity, our field will be the metric itself g_{ab} instead of ψ . Then we define the two auxiliary variables as

$$\Pi_{ab} = -n^c \partial_c g_{ab} = -\frac{1}{\alpha} (\partial_t g_{ab} - \beta^i \partial_i g_{ab}) \quad (1.106)$$

$$\Phi_{iab} = \partial_i g_{ab} \quad (1.107)$$

where $n^c = (\alpha^{-1}, -\alpha^{-1}\beta^i)$ is a timelike unit normal vector, normalized with the spacetime metric in Eq. 1.46. The equation for Φ_{iab} and our gauge condition in Eq. 1.77 give us two constraints

$$\mathcal{C}_a = H_a + \Gamma_a = 0 \quad (1.108)$$

$$\mathcal{C}_{iab} = \partial_i g_{ab} - \Phi_{iab} = 0. \quad (1.109)$$

Rearranging Eq. 1.106, we can get

$$\partial_t g_{ab} - \beta^i \partial_i g_{ab} = -\alpha \Pi_{ab}. \quad (1.110)$$

And if we take the partial derivative of this and rearrange, we can get an equation for $\partial_t \Phi_{iab}$. To account for constraint damping, we add $\gamma_1 \beta^i \mathcal{C}_{iab}$ to the equation for $\partial_t g_{ab}$ and $\gamma_2 \alpha \mathcal{C}_{iab}$ to the equation for $\partial_t \Phi_{iab}$. The equation for $\partial_t \Pi$ is unfortunately somewhat complicated to derive so we omit its derivation, but it will contain terms with $\mathcal{C}_a$ and $\gamma_1 \gamma_2 \beta^i \mathcal{C}_{iab}$ for constraint damping. Instead, we simply present the

first-order symmetric hyperbolic GH system as

$$\partial_t g_{ab} = (1 + \gamma_1) \beta^k \partial_k g_{ab} - \alpha \Pi_{ab} - \gamma_1 \beta^i \Phi_{iab}, \quad (1.111)$$

$$\begin{aligned} \partial_t \Phi_{iab} = & \beta^k \partial_k \Phi_{iab} - \alpha \partial_i \Pi_{ab} + \alpha \gamma_2 \partial_i g_{ab} + \frac{1}{2} \alpha n^c n^d \Phi_{icd} \Pi_{ab} \\ & + \alpha \gamma^{jk} n^c \Phi_{ijc} \Phi_{kab} - \alpha \gamma_2 \Phi_{iab}, \end{aligned} \quad (1.112)$$

$$\begin{aligned} \partial_t \Pi_{ab} = & \beta^k \partial_k \Pi_{ab} - \alpha \gamma^{ki} \partial_k \Phi_{iab} + \gamma_1 \gamma_2 \beta^k \partial_k g_{ab} \\ & + 2\alpha g^{cd} (\gamma^{ij} \Phi_{ica} \Phi_{jdb} - \Pi_{ca} \Pi_{db} - g^{ef} \Gamma_{ace} \Gamma_{bdf}) \\ & - 2\alpha \nabla_{(a} H_{b)} - \frac{1}{2} \alpha n^c n^d \Pi_{cd} \Pi_{ab} - \alpha n^c \Pi_{ci} \gamma^{ij} \Phi_{jab} \\ & + \alpha \gamma_0 (2\delta^c_{(a} n_{b)} - g_{ab} n^c) \mathcal{C}_c - \gamma_1 \gamma_2 \beta^i \Phi_{iab}. \end{aligned} \quad (1.113)$$

Just like the scalar wave system, we can compute the characteristic fields as

$$w_{ab}^g = g_{ab}, \quad (1.114)$$

$$w_{iab}^0 = (\delta_i^k - s^k s_i) \Phi_{kab}, \quad (1.115)$$

$$w_{ab}^\pm = \Pi_{ab} \pm s^i \Phi_{iab} - \gamma_2 g_{ab}, \quad (1.116)$$

along with the characteristic speeds

$$\lambda_{w^g} = -(1 + \gamma_1) \beta^i s_i, \quad (1.117)$$

$$\lambda_{w^0} = -\beta^i s_i, \quad (1.118)$$

$$\lambda_{w^\pm} = \pm \alpha - \beta^i s_i, \quad (1.119)$$

where s^i is the unit normal vector to a 2D surface, the analog of n^k for the scalar wave system. Notice that if $\beta^i s_i = 0$, these characteristic speeds are strikingly similar to the ones in Eqs. 1.96-1.98 for the scalar wave system.

The evolved variables in terms of the characteristic speeds are then

$$g_{ab} = w_{ab}^g, \quad (1.120)$$

$$\Pi_{ab} = \frac{1}{2} (w_{ab}^+ + w_{ab}^-) + \gamma_2 w_{ab}^g, \quad (1.121)$$

$$\Phi_{iab} = \frac{1}{2} (w_{ab}^+ - w_{ab}^-) s_i + w_{iab}^0. \quad (1.122)$$

Again, these are very similar to the scalar wave system, just with the extra spacetime indices.

For GH evolutions, there are two commonly used gauges. The first, harmonic, we have already mentioned

$$H_a = -\Gamma_a = 0. \quad (1.123)$$

A different condition, called damped harmonic, which is used by the SXS Collaboration, is

$$H_a = [\mu_{L1} \log(\sqrt{\gamma}/\alpha) + \mu_{L2} \log(1/\alpha)] n_a - \mu_S g_{ai} \beta^i / \alpha, \quad (1.124)$$

with

$$\mu_{L1} = A_{L1} e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_{L1}}, \quad (1.125)$$

$$\mu_{L2} = A_{L2} e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_{L2}}, \quad (1.126)$$

$$\mu_S = A_S e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_S}. \quad (1.127)$$

Here, r is the 3D distance from the origin and σ_r is a decay width. The damped harmonic gauge condition works to drive both the lapse α and the determinant of the spatial metric $\sqrt{\gamma}$ to one. This avoids two different problems. First, driving the lapse to one ensures that we will have sufficient proper time elapse between each of our spatial hypersurfaces. If the lapse becomes very small, then our evolution will effectively grind to a halt because we will need to take a large number of time steps to not actually advance the simulation very far. Second, driving $\sqrt{\gamma}$ to one avoids exponential growth in the spatial metric when the BHs merge. As the BHs merge, spacetime within the computational domain will become more extreme, and if the metric starts to diverge, we are no longer able to simulate the BHs accurately or stably. The amplitudes and exponents control the amount of damping and the falloff of the gauge condition to the harmonic gauge.

In addition to $\mathcal{C}_a$ and $\mathcal{C}_{iab}$, we are able to define more constraints $\mathcal{C}_{ia}$, $\mathcal{C}_{ijab}$, and $\mathcal{F}_a$. We don't define them here because only $\mathcal{C}_a$ and $\mathcal{C}_{iab}$ are added to the evolution equations to damp away errors. However, they are still useful for monitoring the simulation for errors. We can also combine all of these constraints into a single quantity called the “constraint energy” which gives us an overall notion of how much energy is contained in all of our constraint violations,

$$\begin{aligned} \mathcal{E} = & \delta^{ab} [\mathcal{C}_a \mathcal{C}_b + (\mathcal{F}_a \mathcal{F}_b + \mathcal{C}_{ia} \mathcal{C}_{jb} \gamma^{ij})] \\ & + \delta^{ab} \delta^{cd} (\mathcal{C}_{iac} \mathcal{C}_{jbd} \gamma^{ij} + \mathcal{C}_{ikac} \mathcal{C}_{jld} \gamma^{ij} \gamma^{kl}). \end{aligned} \quad (1.128)$$

Ideally, we'd like zero constraint energy, but numerical error means we must settle for this being as small as possible. Again, we don't go over boundary conditions here, but refer the reader to [10] and [11] for more details.

At this point we will stop our overview of NR. There are still a number of different topics which we did not cover such as (a) initial data on the first spatial hypersurface,

(b) boundary conditions for either BSSN or GH, (c) GW extraction via finite radius, extrapolation, or CCE (see Chapter 4), (d) numerical methods for how to discretize the mesh, (e) methods for taking “good” numerical derivatives, (f) finding the horizon of a black hole (see § 1.7), or (g) any of the previous topics applied to simulating matter around black holes or neutron stars. This field is a rich mix of theory and computational algorithms and deserves to be explored further.

1.5 Goals of NR

Since we now have an idea of how to carry out numerical simulations of black holes, we try to motivate the *why* of it all and what the purpose of NR is. Einstein predicted GWs back in the early 20th century [12, 13]. However, the predictions on the amplitude of the GWs were so small, $\Delta L \sim 10^{-21}\text{m}$, that they were thought to be impossible to measure. Technology eventually advanced sufficiently far that we are now able to measure these GWs here on Earth using lasers in 4km-long vacuum tubes. The first ever signal of two black holes merging was detected by the LIGO [14], VIRGO [15], and Kagra [16] collaborations in 2015 [17, 18].

But when LIGO detected this merger, how did it know it was a merger of two black holes? How did it know that it was a merger at all? There had to be some prediction about what the GWs would look like so data analysts would be able to search the observed data for the predicted signal. These predictions must be based on some sort of theory. However, data analysts don’t know *a priori* what parameters black holes will have when they see a potential signal. They would like to work with theories that give them the parameters of a BH fairly quickly so they can test many predictions. Actually solving Einstein’s equations numerically, though, can be very expensive, so a perturbative theory like Post-Newtonian [19] or Post-Minkowskian [20] is usually better for quick evaluation. These perturbative theories break down when the black holes are close to merger because the perturbations of the background spacetime become too large. The perturbative methods are useful only during the inspiral.

NR exists to fill in the gaps left by the perturbative theories, particularly during merger and ringdown. Fig. 1.6 shows the phases of inspiral, merger, and ringdown from [17]. Plunge happens just before merger. A combination of perturbative theory in the inspiral and NR for merger and ringdown allow for a full theoretical model of a BBH system that LIGO would see. For NR specifically, there exist a number of catalogs of pure NR waveforms for various BBH configurations [21–24].

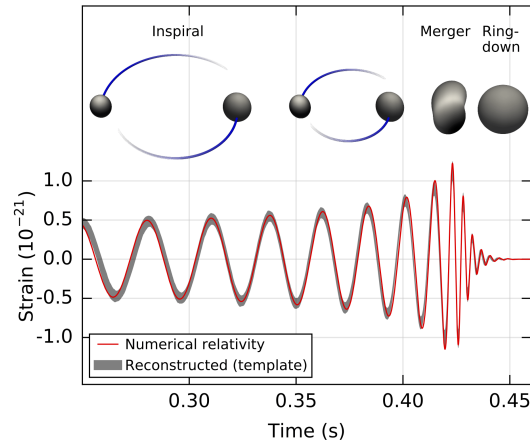


Figure 1.6: The GW150914 signal showing the inspiral, merger, and ringdown of the BBH. Figure taken from [17].

The number of simulations in these catalogs is actually quite low compared to the number needed to fully span the parameter space of LIGO BBHs. This is due to the computational costs of NR simulations (§ 1.6). One method of addressing this issue is to create NR surrogate catalogs which will fill in the BBH parameter space based on existing simulations [25–28]. The other, more popular method is to have an analytic model that is informed by NR. The two common methods are frequency-domain analysis (IMRPhenom family [29–31]) and time-domain analysis (SEOBNR family [32–34]).

Now that LVK has seen hundreds of black hole mergers [18] and done BH population analysis on them [35], there are two natural next steps. First is observing these BBHs with even more precision to better test the theory of GR. Second is expanding the physical systems we are able to detect to include many more binary neutron star systems. Both of these goals, and more, will be accomplished with new, more sensitive, GW detectors. These include Cosmic Explorer [36, 37], Einstein Telescope [38, 39], and the Laser-Interferometer-Space-Antenna (LISA) [40–42], which will all hopefully be operational in the next couple decades.

1.6 Why HPC is Necessary

In order to actually create the NR and GW catalogs, NR simulations must run on supercomputers, or High Performance Computing (HPC) machines, because of the size requirements of the simulations. Here we will break down why one can't run production-ready BBH simulations on a commercial laptop or even desktop.

First, we start with the number of equations we actually need to solve at every single grid point of our domain. For the BSSN formalism this is 24 evolution equations and for the GH formalism this is 50 evolution equations. In addition to the 50 evolved variables, there are typically hundreds (10^2) temporary variables needed to compute the right hand sides of the evolution equations. This includes spatial derivatives and quantities that are reused in multiple equations. This leads to a total of 10^2 - 10^3 floating point values needed.

The number of grid points needed for an evolution varies, but is typically between 10^5 and 10^8 depending on the numerical methods. This means that for a single time step, we could need approximately 10^{10} bytes, or ~ 10 GB, of memory. While this isn't a prohibitive amount of memory, the real problem is performing these calculations in a timely manner. Taking a look back at the BSSN or GH evolution equations, there are many sums that have to be computed for every term in the equations. Trying to run all of these sums on a single core would be incredibly slow. The solution to this is to then parallelize the simulations across many cores, which is where the need for HPC comes from.

Simulations typically take around 10^7 - 10^9 time steps depending on the resolution. If a simulation was parallelized enough to take just 1ms per time step, the entire simulation would take 10^4 - 10^6 s, or 3 hours to 12 days. And this is only if a simulation is able to parallelize efficiently enough. Simulations actually often require weeks to months to run. These numbers are also only for vacuum spacetimes. Coupling a curved spacetime evolution with matter and magnetic fields increases the number of evolved variables and temporaries.

The amount of resources used on HPCs also varies based on the numerical methods used. These are often measured in core hours, which is the number of hours used times the number of cores used. The SXS Collaboration's recent catalog of 3756 NR waveforms cost approximately 4.8×10^8 core hours[43], while a recent paper [44] cited just 4 high resolution NR waveforms costing approximately 1.55×10^8 core hours. This difference can likely be attributed to the use of spectral methods vs. finite-difference methods. In recent years, advancements in GPU architecture have made simulations of BBHs and neutron stars possible with GPUs [45–51]; however, there is still much to be done in this area.

1.7 Horizon Finding in the GH Formalism

Here, we will only discuss horizon finding as it pertains to the GH formalism. Horizons can be found in the BSSN formulation as well using similar techniques.

Finding horizons in the GH formalism is necessary for the evolution of the equations in a way that it isn't for BSSN. In BSSN, the physical singularities are removed from the spacetime and replaced with punctures that approximate the singularities. This works because of some details of the gauge conditions used in the BSSN formalism and because information can never leave a BH. However, in GH, the harmonic (or damped harmonic) gauge conditions don't allow this treatment, so there are two physical singularities for a BBH system. At these singularities, the metric becomes singular so we must avoid putting any grid points there.

To avoid the singularity, simulations of GH typically employ the excision method, where a portion of the domain around the singularity is removed and not simulated. This cavity, called the excision, avoids the need for resolving the spacetime metric near the singularity, but now introduces a new “outer” boundary to our domain. And a new outer boundary implies that there must be a new boundary condition.

Before we discuss the new boundary condition, we first briefly describe the algorithm for finding apparent horizons (AHs). An apparent horizon approximates the 2D surface of an event horizon of a BH at a specific coordinate time, but is not exactly equal to the event horizon. This is because an event horizon is the surface where no outward-pointing null rays will reach future null infinity ($\mathcal{I}^+$). However, to know whether a null ray reaches $\mathcal{I}^+$, one must know the entire future of the spacetime. We obviously don't have the entire future spacetime while trying to simulate spatial slices, so we approximate the event horizon with the AH.

The process of AH finding is outlined as follows. For more details, see [52]. We start off with some closed 2-surface $\mathcal{S}$ embedded in our spatial hypersurface Σ , as shown in Fig. 1.7. We define n^a in the usual way as the timelike normal to Σ . We define the spatial (and spacelike) normal to the surface $\mathcal{S}$ as s^a . Using n^a and s^a , we can define three new quantities

$$m_{ab} = \gamma_{ab} - s_a s_b = g_{ab} + n_a n_b - s_a s_b \quad (1.129)$$

$$\ell^a = \frac{1}{\sqrt{2}}(n^a - s^a) \quad (1.130)$$

$$k^a = \frac{1}{\sqrt{2}}(n^a + s^a) \quad (1.131)$$

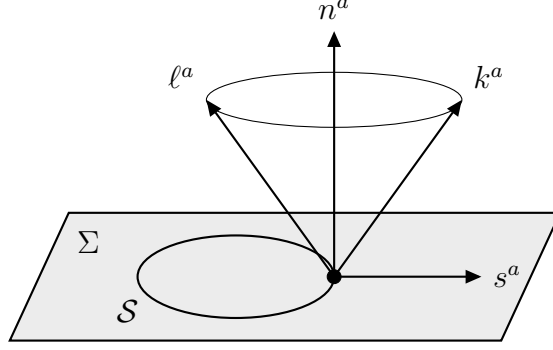


Figure 1.7: A closed 2-surface $\mathcal{S}$ embedded in a spacelike hypersurface Σ . The vectors ℓ^a and k^a are the ingoing and outgoing null normals to $\mathcal{S}$, respectively, constructed from the timelike unit normal n^a to Σ and the outward spacelike unit normal s^a to $\mathcal{S}$ within Σ .

where m_{ab} is the metric on the 2-surface constructed from the spatial metric γ_{ab} and ℓ^a and k^a are the ingoing and outgoing null vectors from point P on the surface.

We define the expansion of $\mathcal{S}$ as

$$\Theta = g^{ab} \nabla_a k_b = \nabla_a s^a - K + s^a s^b K_{ab} \quad (1.132)$$

where K_{ab} and K are the extrinsic curvature and its trace. We define $\Theta = 0$ as a marginally outer-trapped surface, meaning all null rays are ingoing. With some point contained within $\mathcal{S}$ denoted as c^i , we also define level surfaces as

$$\tau = \sqrt{\delta_{ij}(x^i - c^i)(x^j - c^j)} - r(\theta, \phi) = C \quad (1.133)$$

for every point x^i on $\mathcal{S}$. Here, C is a constant. We can then define the surface normal form $s_a = (0, s_i)$ as

$$s_i = \frac{\partial_i \tau}{|s|}, \quad |s| = \sqrt{\delta_{ij}(\partial_i \tau)(\partial_j \tau)}. \quad (1.134)$$

We can then plug this into Eq. 1.132 to get

$$\Theta = (\gamma^{ij} - s^i s^j) \left(\frac{|s|}{r} (\delta_{ij} - \hat{r}_i \hat{r}_j) - \lambda \partial_i \partial_j \Gamma - s_k \Gamma_{ij}^k - K_{ij} \right) \quad (1.135)$$

with

$$\hat{r}^i = \frac{x^i - c^i}{\sqrt{\delta_{kj}(x^k - c^k)(x^j - c^j)}}, \quad s_i = \frac{1}{|s|} (\hat{r}_i - \partial_i r), \quad (1.136)$$

$$\hat{r}_i = \delta_{ij} \hat{r}^j, \quad s^i = \gamma^{ij} s_j. \quad (1.137)$$

Our goal is to find the $r(\theta, \phi)$ function describing the surface $\mathcal{S}$ such that $\tau = 0$. The quantities γ^{ij} , K_{ij} , and Γ_{ij}^k are needed at the points of $r(\theta, \phi)$.

The most popular method to find $r(\theta, \phi)$ is the fast-flow algorithm by [52]. We can define r as

$$r(\theta, \phi) = \sum_{\ell, m}^{L_{\max}} a_{\ell m} Y_{\ell m}(\theta, \phi). \quad (1.138)$$

Then, via some derivation we won't show here, one can show there is a recursion relation between successive iterations of the coefficients $a_{\ell m}$ as

$$a_{\ell m}^{(n+1)} - a_{\ell m}^{(n)} = \frac{-A}{1 + B\ell(\ell + 1)} (\rho\Theta)_{\ell m}^{(n)} \quad (1.139)$$

where

$$A = \frac{1}{L_{\max}(L_{\max} + 1)} + \frac{1}{2}, \quad (1.140)$$

$$B = \frac{1}{2}, \quad (1.141)$$

$$\rho = \frac{2|s|r^2}{(\gamma^{ij} - s^i s^j)(\delta_{ij} - \hat{r}_i \hat{r}_j)}. \quad (1.142)$$

With this algorithm, one must supply an initial guess for $a_{\ell m}^{(0)}$, which is usually just a sphere. The algorithm continues until the expansion Θ is below some tolerance set by the user.

An AH is only an approximation of the event horizon. However, the AH will always be contained entirely within the event horizon. This arises due to the definition of an AH. Specifically, that it is a marginally outer-trapped surface contained within a spatial hypersurface at one specific time. At the surface of the AH, all null rays are ingoing. However, for an event horizon, there will be a region where some null rays are outgoing on a specific spatial hypersurface, but they cannot escape to $\mathcal{I}^+$ and will eventually become ingoing null rays at a later spatial hypersurface. This region in the first hypersurface will be outside the AH (because the null rays are outgoing), but still inside the event horizon, thus showing that the AH is within the event horizon. For the Kerr (and Schwarzschild spacetime) the AH and the event horizon at a specific spatial hypersurface are the same.

We return now to the boundary condition that must be imposed on the excision boundary. We will make use of the fact that the AH will always be contained within the event horizon. We will mandate that the excision surface always be within the AH of the BH we are simulating, therefore all information (null rays) at the excision

surface must be ingoing (i.e. towards the singularity). With this requirement, we need not impose any boundary conditions at the excision boundary. This is because there is not any information coming into the simulation (i.e. outgoing null rays), therefore we don't need to prescribe how information enters the domain if none is entering.

The challenge then becomes ensuring that the excision surface stays inside the AH for the entire simulation of two BHs inspiraling and eventually merging. More details about how that is done can be found in Ch. 2.

1.8 Overview of Chapters

Here we give an overview of the remaining chapters of the thesis.

1.8.1 Chapter 2

Ch. 2 is the published work [53]. It expands significantly on § 1.7, detailing how horizon finding is done in the SpECTRE code. Specifically, it goes into detail about how the asynchronous nature of SpECTRE posed a new challenge for designing the dynamic feedback control system that keeps the excisions within the horizons, and the same shape as the horizons. It explains our solution to these new challenges and gives examples of the control system being used in scientific examples, including a binary neutron star system.

1.8.2 Chapter 3

Ch. 3 is the published work [54]. It uses the implementation of the control system and horizon finder presented in Ch. 2 to perform the first BBH inspiral, merger, and ringdown simulation using discontinuous Galerkin spectral methods for the spatial discretization. The chapter starts by performing some test cases on single BH and gauge wave problems showing convergence and stability. Then, three different resolutions of an equal-mass non-spinning BBH system are presented along with analysis of the constraint violations and numerical waveforms extracted with SpECTRE CCE.

1.8.3 Chapter 4

Ch. 4 is not directly in any existing published work. It describes a new feature of the SpECTRE CCE module that is able to convert worldtube data from other NR codes besides SpEC and SpECTRE into the format that SpECTRE requires. This includes computing time derivatives of the spatial metric, lapse, and shift based on

the gauge conditions used for the evolution, whether it be BSSN or some Z4 variant. In addition to the time derivatives, the new feature can also read in the necessary quantities as either nodal values (i.e. values at grid points) or modal coefficients (i.e. coefficients in a spin-weighted spherical harmonic expansion). This gives the user of SpECTRE CCE the flexibility to output worldtube data from their code in whatever format is easiest for them. This feature has already been used in a published work for the AthenaK code [55], and a soon-to-be-published work for the Einstein Toolkit [56, 57].

Bibliography

- [1] Thomas W. Baumgarte and Stuart L. Shapiro. *Numerical Relativity: Solving Einstein's Equations on the Computer*. Cambridge University Press, 2010. doi: 10.1017/CBO9781139193344.
- [2] Masaru Shibata and Takashi Nakamura. Evolution of three-dimensional gravitational waves: Harmonic slicing case. *Phys. Rev. D*, 52:5428–5444, 1995. doi: 10.1103/PhysRevD.52.5428.
- [3] Thomas W. Baumgarte and Stuart L. Shapiro. On the numerical integration of Einstein's field equations. *Phys. Rev. D*, 59:024007, 1998. doi: 10.1103/PhysRevD.59.024007.
- [4] Richard L. Arnowitt, Stanley Deser, and Charles W. Misner. Republication of: The dynamics of general relativity. *Gen. Rel. Grav.*, 40:1997–2027, 2008. doi: 10.1007/s10714-008-0661-1.
- [5] Carles Bona, Tomáš Ledvinka, Carlos Palenzuela, and Michal Žáček. General covariant evolution formalism for numerical relativity. *Phys. Rev. D*, 67:104005, 2003. doi: 10.1103/PhysRevD.67.104005.
- [6] Sebastiano Bernuzzi and David Hilditch. Constraint violation in free evolution schemes: Comparing BSSNOK with a conformal decomposition of Z4. *Phys. Rev. D*, 81:084003, 2010. doi: 10.1103/PhysRevD.81.084003.
- [7] David Hilditch, Sebastiano Bernuzzi, Marcus Thierfelder, Zhoujian Cao, Wolfgang Tichy, and Bernd Bruegmann. Compact binary evolutions with the Z4c formulation. *Phys. Rev. D*, 88:084057, 2013. doi: 10.1103/PhysRevD.88.084057.
- [8] Daniela Alic, Carles Bona-Casas, Carles Bona, Luciano Rezzolla, and Carlos Palenzuela. Conformal and covariant formulation of the Z4 system with constraint-violation damping. *Phys. Rev. D*, 85:064040, 2012. doi: 10.1103/PhysRevD.85.064040.

- [9] Lawrence E. Kidder, Mark A. Scheel, and Saul A. Teukolsky. Extending the lifetime of 3D black hole computations with a new hyperbolic system of evolution equations. *Phys. Rev. D*, 64:064017, 2001. doi: 10.1103/PhysRevD.64.064017.
- [10] Lee Lindblom, Mark A. Scheel, Lawrence E. Kidder, Robert Owen, and Oliver Rinne. A new generalized harmonic evolution system. *Class. Quant. Grav.*, 23:S447–S462, 2006. doi: 10.1088/0264-9381/23/16/S09.
- [11] Oliver Rinne, Lee Lindblom, and Mark A. Scheel. Testing outer boundary treatments for the Einstein equations. *Class. Quant. Grav.*, 24:4053–4078, 2007. doi: 10.1088/0264-9381/24/16/006.
- [12] Albert Einstein. Approximative integration of the field equations of gravitation. *Sitzungsber. Preuss. Akad. Wiss. Berlin (Math. Phys.)*, 1916:688–696, 1916.
- [13] Albert Einstein. Über Gravitationswellen. *Sitzungsber. Preuss. Akad. Wiss. Berlin (Math. Phys.)*, 1918:154–167, 1918.
- [14] Junaid Aasi et al. Advanced LIGO. *Class. Quant. Grav.*, 32:074001, 2015. doi: 10.1088/0264-9381/32/7/074001.
- [15] Fausto Acernese et al. Advanced Virgo: A second-generation interferometric gravitational wave detector. *Class. Quant. Grav.*, 32(2):024001, 2015. doi: 10.1088/0264-9381/32/2/024001.
- [16] Tomotada Akutsu et al. Overview of KAGRA: Detector design and construction history. *PTEP*, 2021(5):05A101, 2021. doi: 10.1093/ptep/ptaa125.
- [17] Benjamin P. Abbott et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116(6):061102, 2016. doi: 10.1103/PhysRevLett.116.061102.
- [18] Robert Abbott et al. GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run. *Phys. Rev. X*, 13(4):041039, 2023. doi: 10.1103/PhysRevX.13.041039.
- [19] Luc Blanchet. Gravitational radiation from post-Newtonian sources and inspiralling compact binaries. *Living Rev. Rel.*, 17:2, 2014. doi: 10.12942/lrr-2014-2.
- [20] Zvi Bern, Clifford Cheung, Ira Z. Solon, et al. Black hole binary dynamics from the double copy and effective theory. *JHEP*, 10:206, 2019. doi: 10.1007/JHEP10(2019)206.
- [21] Michael Boyle et al. The SXS collaboration catalog of binary black hole simulations. *Class. Quant. Grav.*, 36(19):195006, 2019. doi: 10.1088/1361-6382/ab34e2.

- [22] James Healy and Carlos O. Lousto. Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits. *Phys. Rev. D*, 105(12):124010, 2022. doi: 10.1103/PhysRevD.105.124010.
- [23] Deborah Ferguson et al. Second MAYA catalog of binary black hole numerical relativity waveforms. *Phys. Rev. D*, 112(4):044043, 2025. doi: 10.1103/gk7x-9cds.
- [24] Eleanor Hamilton et al. Catalog of precessing black-hole-binary numerical-relativity simulations. *Phys. Rev. D*, 109(4):044032, 2024. doi: 10.1103/PhysRevD.109.044032.
- [25] Jonathan Blackman, Scott E. Field, Mark A. Scheel, Chad R. Galley, Christian D. Ott, Michael Boyle, Lawrence E. Kidder, Harald P. Pfeiffer, and Bela Szilagyi. A numerical relativity waveform surrogate model for generically precessing binary black hole mergers. *Phys. Rev. D*, 96:024058, 2017. doi: 10.1103/PhysRevD.96.024058.
- [26] Vijay Varma, Scott E. Field, Mark A. Scheel, Jonathan Blackman, Lawrence E. Kidder, and Harald P. Pfeiffer. Surrogate model of hybridized numerical relativity binary black hole waveforms. *Phys. Rev. D*, 99:064045, 2019. doi: 10.1103/PhysRevD.99.064045.
- [27] Vijay Varma, Scott E. Field, Mark A. Scheel, Jonathan Blackman, Davide Gerosa, Leo C. Stein, Lawrence E. Kidder, and Harald P. Pfeiffer. Surrogate models for precessing binary black hole simulations with unequal masses. *Phys. Rev. Research*, 1:033015, 2019. doi: 10.1103/PhysRevResearch.1.033015.
- [28] Lorena Magaña Zertuche et al. High-precision ringdown surrogate model for nonprecessing binary black holes. *Phys. Rev. D*, 112(2):024077, 2025. doi: 10.1103/q7sy-g3kl.
- [29] Geraint Pratten, Sascha Husa, Cecilio Garcia-Quiros, Marta Colleoni, Antoni Ramos-Buades, Hector Estelles, and Rafel Jaume. Setting the cornerstone for a family of models for gravitational waves from compact binaries: The dominant harmonic for nonprecessing quasicircular black holes. *Phys. Rev. D*, 102(6):064001, 2020. doi: 10.1103/PhysRevD.102.064001.
- [30] Cecilio García-Quirós, Marta Colleoni, Sascha Husa, Héctor Estellés, Geraint Pratten, Antoni Ramos-Buades, Maite Mateu-Lucena, and Rafel Jaume. Multimode frequency-domain model for the gravitational wave signal from nonprecessing black-hole binaries. *Phys. Rev. D*, 102(6):064002, 2020. doi: 10.1103/PhysRevD.102.064002.
- [31] Geraint Pratten et al. Computationally efficient models for the dominant and subdominant harmonic modes of precessing binary black holes. *Phys. Rev. D*, 103(10):104056, 2021. doi: 10.1103/PhysRevD.103.104056.

- [32] Mohammed Khalil, Alessandra Buonanno, Hector Estelles, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, and Antoni Ramos-Buades. Theoretical groundwork supporting the precessing-spin two-body dynamics of the effective-one-body waveform models SEOBNRv5. *Phys. Rev. D*, 108(12): 124036, 2023. doi: 10.1103/PhysRevD.108.124036.
- [33] Antoni Ramos-Buades, Alessandra Buonanno, Héctor Estellés, Mohammed Khalil, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, and Mahlet Shiferaw. Next generation of accurate and efficient multipolar precessing-spin effective-one-body waveforms for binary black holes. *Phys. Rev. D*, 108(12): 124037, 2023. doi: 10.1103/PhysRevD.108.124037.
- [34] Maarten van de Meent, Alessandra Buonanno, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, Niels Warburton, Adam Pound, Barry Wardell, Leanne Durkan, and Jeremy Miller. Enhancing the SEOBNRv5 effective-one-body waveform model with second-order gravitational self-force fluxes. *Phys. Rev. D*, 108(12):124038, 2023. doi: 10.1103/PhysRevD.108.124038.
- [35] Robert Abbott et al. Population of merging compact binaries inferred using gravitational waves through GWTC-3. *Phys. Rev. X*, 13(1):011048, 2023. doi: 10.1103/PhysRevX.13.011048.
- [36] Matthew Evans et al. A horizon study for Cosmic Explorer: Science, observatories, and community. September 2021. URL <https://arxiv.org/abs/2109.09882>.
- [37] Matthew Evans et al. Astrophysics with Cosmic Explorer. June 2023. URL <https://arxiv.org/abs/2306.16219>.
- [38] Michele Maggiore et al. Science case for the Einstein Telescope. *JCAP*, 03: 050, 2020. doi: 10.1088/1475-7516/2020/03/050.
- [39] Marica Branchesi et al. Science with the Einstein Telescope: A comparison of different designs. *JCAP*, 07:068, 2023. doi: 10.1088/1475-7516/2023/07/068.
- [40] Pau Amaro-Seoane et al. Laser interferometer space antenna. February 2017. URL <https://arxiv.org/abs/1702.00786>.
- [41] Pau Amaro-Seoane et al. Astrophysics with the laser interferometer space antenna. *Living Rev. Rel.*, 26(1):2, 2023. doi: 10.1007/s41114-022-00041-y.
- [42] Monica Colpi et al. LISA definition study report. February 2024. URL <https://arxiv.org/abs/2402.07571>.
- [43] Mark A. Scheel et al. The SXS collaboration’s third catalog of binary black hole simulations. *Class. Quant. Grav.*, 42(19):195017, 2025. doi: 10.1088/1361-6382/adfd34.

- [44] Alireza Rashti, Rossella Gamba, Koustav Chandra, David Radice, Boris Daszuta, William Cook, and Sebastiano Bernuzzi. Binary black hole waveforms from high-resolution GR-Athena++ simulations. *Phys. Rev. D*, 111(10):104078, 2025. doi: 10.1103/n5pz-qv3x.
- [45] Adam J. Peterson, Don Willcox, Philipp Mösta, and Phillip Moesta. Code generation for AMReX with applications to numerical relativity. *Class. Quant. Grav.*, 40(24):245013, 2023. doi: 10.1088/1361-6382/ad0b37.
- [46] Swapnil Shankar, Philipp Mösta, Steven R. Brandt, Roland Haas, Erik Schnetter, and Yannick de Graaf. GRaM-X: A new GPU-accelerated dynamical spacetime GRMHD code for exascale computing with the Einstein Toolkit. *Class. Quant. Grav.*, 40(20):205009, 2023. doi: 10.1088/1361-6382/acf2d9.
- [47] Jay V. Kalinani et al. AsterX: A new open-source GPU-accelerated GRMHD code for dynamical spacetimes. *Class. Quant. Grav.*, 42(2):025016, 2025. doi: 10.1088/1361-6382/ad9c11.
- [48] Hengrui Zhu, Jacob Fields, Francesco Zappa, David Radice, James M. Stone, Alireza Rashti, William Cook, Sebastiano Bernuzzi, and Boris Daszuta. Performance-portable numerical relativity with AthenaK. *Astrophys. J. Suppl.*, 278(2):50, 2025. doi: 10.3847/1538-4365/adcf96.
- [49] Jacob Fields, Hengrui Zhu, David Radice, James M. Stone, William Cook, Sebastiano Bernuzzi, and Boris Daszuta. Performance-portable binary neutron star mergers with AthenaK. *Astrophys. J. Suppl.*, 276(2):35, 2025. doi: 10.3847/1538-4365/ad9687.
- [50] Liwei Ji, Roland Haas, Yosef Zlochower, Steven R. Brandt, Erik Schnetter, and Allen Wen. GPU-accelerated subcycling time integration with the Einstein Toolkit. *Phys. Rev. D*, 112(2):024049, 2025. doi: 10.1103/bqpq-5cp9.
- [51] Samuel D. Tootle, Leonardo R. Werneck, Thiago Assumpção, Terrence Pierre Jacques, and Zachariah B. Etienne. Accelerating numerical relativity with code generation: CUDA-enabled hyperbolic relaxation. *Class. Quant. Grav.*, 42(11):115005, 2025. doi: 10.1088/1361-6382/add63e.
- [52] Carsten Gundlach. Pseudospectral apparent horizon finders: An efficient new algorithm. *Phys. Rev. D*, 57:863–875, 1998. doi: 10.1103/PhysRevD.57.863.
- [53] Kyle C. Nelli, William Throwe, Nils Deppe, Mark A. Scheel, Lawrence E. Kidder, Nils L. Vu, and Saul A. Teukolsky. Horizon tracking for asynchronous parallel black hole simulations. August 2025. URL <https://arxiv.org/abs/2508.08408>.
- [54] Geoffrey Lovelace, Kyle C. Nelli, et al. Simulating binary black hole mergers using discontinuous Galerkin methods. *Class. Quant. Grav.*, 42(3):035001, 2025. doi: 10.1088/1361-6382/ad9f19.

- [55] David Radice, Rossella Gamba, Hengrui Zhu, and Alireza Rashti. AthenaK simulations of the binary black hole merger GW150914. *Class. Quant. Grav.*, 42(18):185003, 2025. doi: 10.1088/1361-6382/adfffa.
- [56] Frank Löffler et al. The Einstein Toolkit: A community computational infrastructure for relativistic astrophysics. *Class. Quant. Grav.*, 29:115001, 2012. doi: 10.1088/0264-9381/29/11/115001.
- [57] Maxwell Rizzo, Roland Haas, Steven R. Brandt, et al. The Einstein Toolkit (the ‘martin d. Kruskal’ release, et_2025_05), 2025. URL <https://zenodo.org/records/15520463>.

Chapter 2

HORIZON TRACKING FOR ASYNCHRONOUS PARALLEL BLACK HOLE SIMULATIONS

Kyle C. Nelli, William Throwe, Nils Deppe, Mark A. Scheel, Lawrence E. Kidder, Nils L. Vu, and Saul A. Teukolsky. Horizon tracking for asynchronous parallel black hole simulations. August 2025. URL <https://arxiv.org/abs/2508.08408>.

In the field of gravitational wave science, next-generation detectors will be substantially more accurate than the current suite of detectors. Numerical relativity simulations of binary black hole (BBH) gravitational waveforms must become faster, more efficient, and more accurate to be used in analyses of these next-generation detections. One approach, which the SpECTRE code employs, is using spectral methods for accuracy along with asynchronous task-based parallelism to avoid idle time in simulations and make the most efficient use of computational resources. When writing an asynchronous application, algorithms must be redesigned compared to their synchronous counterparts. To illustrate this process, we present novel methods for dynamically tracking the apparent horizons in evolutions of BBH mergers using a feedback control system, all in the context of asynchronous parallelism. This algorithm enables local time stepping with spectral methods for BBH simulations, which reduces the computational cost of a mass ratio 4 BBH simulation by a factor of five. Further reductions in computational cost are expected for higher mass ratios and spinning binaries. We also briefly detail how these methods can be applied to binary neutron star simulations performed with asynchronous parallelism.

2.1 Introduction

Gravitational waves provide a new window into our universe. In order to observe these gravitational waves, detectors such as LIGO [1], Virgo [2], KAGRA [3], and GEO600 [4] have been built and in 2015, the first gravitational waves from the merger of a binary black hole (BBH) system were observed [5]. Since the first detection, there have been several hundred detections of compact binary systems, including BBHs, binary neutron stars, and binary black-hole neutron-star systems [6]. To compare these observations to theory, one must solve Einstein’s equations of general relativity using numerical relativity (NR), which is needed because of the lack of a closed-form solution to the two-body problem in general relativity. A number of different NR codes have been developed to simulate compact object mergers. Furthermore, several publicly available catalogs of gravitational waveforms have been released for use in comparing to observations [7–15].

In order to expand the number of detections and to perform more accurate observations, there are plans to build “next-generation” gravitational wave detectors such as the Laser Interferometer Space Antenna (LISA) [16], the Einstein Telescope [17], and Cosmic Explorer [18]. While the current NR catalogs of gravitational waveforms are sufficient for understanding observations from the current set of detectors, they are inadequate for analyzing observations from these next-generation detectors. The gravitational waveforms that are needed must be at least $10\times$ more accurate than the current catalogs [19–21].

To solve this problem, the NR community has been developing new or improved codes to meet the accuracy requirements for next-generation detectors. Some of these codes are SpEC [22], SpECTRE [23], AthenaK [24], and codes using the Einstein Toolkit with the CarpetX driver such as GRaM-X [25] and AsterX [26]. The approaches of these new codes can be split into two broad categories: 1) Finite-difference methods run on GPUs and 2) Spectral methods run on CPUs. Most codes fall into the first category, taking advantage of the power of GPUs to perform a large number of computations very quickly using large chunks of the computational domain. Finite-difference methods are very robust, making them appealing for the long simulations needed in numerical relativity. Finite-difference methods typically use a cartesian grid, which lends itself very well to GPUs. The new CarpetX driver implemented in the Einstein Toolkit is built on top of the AMReX [27] adaptive mesh refinement infrastructure. AthenaK makes use of the Kokkos framework [28] for performance-portable simulations so the user can focus on the equations rather

than the parallel infrastructure.

A few codes, such as SpECTRE, SpEC, and NMesh [29], fall into the second category and take advantage of the computational efficiency of spectral methods over finite-difference methods. SpECTRE is distinct from SpEC and NMesh because it uses asynchronous parallelism, while the others use some form of synchronous parallelism. The goal of asynchronous parallelism is to have as little idle time as possible, no matter the number of resources you run on.

SpECTRE uses local time stepping (LTS) for integrating the evolution equations [30]. However, in order to use LTS, all other aspects of the simulation must be as asynchronous as possible. If not, then the largest allowed time step size may be limited by, e.g., how often the apparent horizons need to be found for moving the excision surfaces. Thus, the horizon finding and control system update algorithms used in BBH simulations employing spectral methods should be asynchronous, which they are not in SpEC. There is not one unique asynchronous horizon tracking algorithm and we present the one that we arbitrarily decided to implement. We show that the algorithm works with LTS and reduces the computational cost of a mass ratio 4 BBH simulation by 5x. The savings are expected to increase for higher mass ratios and rapidly spinning black holes. Using LTS in excision-based BBH simulations is the primary use-case for the algorithm described here. While using finite-difference methods with punctures to avoid excision seems simpler, spectral methods are thousands of times more efficient than state-of-the-art finite-difference codes [12].

The paper is structured as follows: In § 2.2.1 we introduce the concept of asynchronous parallelism and how it differs from the synchronous case. We illustrate with a straightforward example the difference between the two using input/output (I/O). In § 2.3, we detail how to implement a horizon finder in the context of asynchronous parallelism. In § 2.4, we introduce the synchronous aspects of the feedback control system in SpECTRE along with some necessary context about the domain and time-dependent coordinate mappings we use. Then in § 2.5 we detail changes necessary to implement the feedback control system with asynchronous parallelism, along with a number of subtleties that arise. § 2.6 provides performance and timing results demonstrating the efficiency improvement gained from enabling LTS. In § 2.7 we briefly explain how the feedback control system can be applied to binary neutron star simulations done with asynchronous parallelism in SpECTRE. And finally, we summarize and conclude our findings in § 2.8.

2.2 Asynchronous Parallelism

In this section, we first introduce the concept of asynchronous parallelism and how it differs from a synchronous application. It is important to make the differences explicit as they will shape how we think about and structure our asynchronous algorithm.

2.2.1 Difference from Synchronous Parallelism

Synchronous parallel applications are typically implemented using the widely-used Message Passing Interface (MPI), where the problem is divided as evenly as possible among MPI ranks. Typically, there is one rank per core, with each rank running the same application in parallel. For all but the most computationally-trivial problems, the ranks are not completely independent. Instead, ranks must receive messages from each other. This is accomplished using the various `MPI_Send`/`MPI_Receive` calls to send and receive messages between the MPI ranks at specific pre-determined stages during the computation. Also, each rank typically pauses at each of these message-exchange stages, continuing execution only after all the ranks have exchanged messages.

Another method for implementing synchronous parallel applications is to instead have only one MPI rank per node. The rest of the cores on the node are then used for OpenMP acceleration in `for` loops. While this does differ from the typical MPI application because not every core runs the same application, there's still one core on a node which runs the same application and is strongly synchronized with other ranks. Therefore, this method is still synchronous, but may have better performance.

Asynchronous parallelism (also called task-based parallelism or asynchronous many-task parallelism) on the other hand, is a type of parallelism that makes use of small “tasks” that are launched on any available resource from a master process. A task can be thought of as a function that runs on some resource, can edit data that is stored in memory, and usually doesn't return a value. Several third-party libraries such as Uintah [31], Chapel [32], Charm++ [33], Kokkos [28], Legion [34], and PaRSEC [35] implement task-based parallelism in slightly different ways. Often asynchronous parallelism is implemented utilizing MPI with threading and asynchronous send/receive calls. Therefore, the “driver process” is typically an MPI rank and the “available resources” are the threads managed by that MPI rank. In our application SpECTRE we choose to use Charm++, but we are not locked into that choice. The methods presented here will be applicable to any asynchronous runtime

system.

One of the critical differences between asynchronous and synchronous applications is the guarantee of message ordering. According to the MPI standard [36], most synchronous messages are *non-overtaking*, meaning that if a single sender sends two messages A and B in order to the same destination, they will be received in the same order (A then B). Non-overtaking messages allow for deterministic message ordering in an application and a guarantee of synchronicity. In asynchronous runtime systems, however, this relatively simple requirement is no longer guaranteed. This fact is fundamental enough to the following algorithms to give it a formal statement:

Statement 1. *If a single sender sends successive messages A then B to the same receiver, there is **no** guarantee which message will arrive first at the receiver.*

The consequences of statement 1 are subtle, yet cascading. Applications that were originally written to be synchronous must typically be completely rewritten to make them suitable for an asynchronous algorithm. Communication, I/O, and global synchronizations must all be rethought.

Lastly, we define some terminology that was used in this section and will be reused throughout this paper. As stated before, a *task* can be thought of as a function that typically doesn't return a value. This task will typically be run on a *thread*, a process of the application, and that thread will be assigned to a *core* of a CPU. For this reason, we typically use thread and core interchangeably even though they are technically different. We then have a *node* which is a collection of cores. Finally, we'll define the concept of an *algorithm* to be a collection of tasks and associated data stored in memory that these tasks will have access to. An example of an algorithm is described in the next section.

2.2.2 I/O Example

To give a straightforward example of an algorithm that needs to be rethought when using task-based parallelism, let's consider I/O from our computation. Imagine that we have data distributed across our computational resources that depends on simulation time. We would like to write the data to a file on disk at each simulation time. (For simplicity, in the remainder of this paper we will use "time" to refer to simulation time. When we need to refer to wallclock time, we will do so explicitly.) Suppose that one MPI rank is dedicated to writing the data to disk. In a synchronous implementation (we'll use SpEC as an example), each rank would compute the

necessary data to write to disk at a given time and wait for all other ranks to finish at that same time. Then, all ranks send their data synchronously to the dedicated writing rank, the data is combined, and written out to disk. Once writing is complete, each rank can continue the application and advance to the next time, combine the new data, and write it to disk. The drawback to this method is the scenario when a rank finishes its work quickly and could potentially move along to the next time and send data for I/O at this new time, but can't because it must wait to participate in the first time's reduction.

Now we consider I/O in an asynchronous implementation with the same setup, replacing ranks with tasks and the writing rank with a passive thread that will run tasks which actually write the data to disk. All tasks start computing their data that will be written to disk, and some finish more quickly than others. Because this is an asynchronous implementation, a task that finishes quickly doesn't need to wait for other tasks to finish. It sends its own portion of the data and launches a task on the dedicated writing thread and is now free to continue doing work. This freedom to keep doing additional work without synchronizing with any other tasks can improve efficiency, but must be handled carefully because race conditions could arise.

Consider a case where tasks run on a single node and some tasks finish faster than others. A direct example of Statement 1 would be that a task A is able to do so much work that it sends two separate messages to the dedicated writing thread for different times before task B even sends its first message. This possibility implies that an I/O algorithm must not only be able to accept data from tasks at a single time in any order, but should also be able to accept data at any time because it may arrive "out of order." This will add code complexity to an I/O algorithm and can (and will) inform how data is stored in memory and on disk to accommodate receiving portions of the data out of temporal order. We don't detail a specific implementation on how to store this data because it is application dependent, but we do suggest that the solution involves 1) knowing how many tasks will be sending data to the writing thread so the I/O algorithm knows when it has received all data and 2) storing the data in a map-like data structure sorted by time.

2.2.3 Complications with I/O Example

It is worth noting the potential issues that can arise if an asynchronous algorithm is implemented incorrectly, as they can cause issues that one may not expect. The I/O example is a passive asynchronous algorithm, meaning that there is no feedback

from the writing thread onto the rest of the application, so we don't have to worry about deadlocks (cf. § 2.5.6) here. However, an issue can arise if the number of tasks contributing data to I/O at each time is smaller than the number of messages the writing thread expects to receive. In this scenario, data will continue to be sent to the writing thread, but it will never be written to disk and purged from memory, causing a “memory leak”. “Memory leak” is in quotes because it is not the typical memory leak found in computer science where a pointer's memory is not freed correctly, but is rather an algorithmic bug that causes data to be stored for too long. This can eventually result in an out-of-memory (OOM) error.

Another possible issue can occur even if the I/O algorithm is implemented correctly. Consider a case where some tasks are able to advance far ahead in time and continually send data to the writing thread, while other tasks lag behind and don't send any data (possibly their workload is much larger). A similar OOM issue to the previous one can arise where the writing thread keeps storing more and more data without writing to disk since it has not received data from all tasks at a given time. However, now the OOM issue is not due to a bug, but is due to the (possibly random) unequal workload among tasks. This must be dealt with in some way, possibly by limiting how far ahead in time tasks can proceed, or having the writing thread write whatever partial data it has to disk when it reaches some threshold memory usage.

2.3 Apparent Horizon Finder

We start by detailing how a standalone apparent horizon finding algorithm can be implemented in an asynchronous application. Specifically, we'll describe what is implemented in SpECTRE. For the synchronous part of the horizon finding algorithm, we use a fast-flow method similar to [37]. This is an iterative method that takes as input a trial surface, and at each iteration it produces a new trial surface that is closer to the actual apparent horizon. However, the details of the algorithm are unimportant for the discussion here as it does not affect the parallelism. What is important is that for each iteration of the method, evolution variables in the volume (for example the spatial metric and extrinsic curvature) must be interpolated onto the trial surface. The volume that is evolved is usually split into smaller partitions we'll call cells. For a synchronous implementation, the horizon finding algorithm is relatively straightforward:

1. All cells finish their time step. Evolution variables are now in a consistent state at the same time.

2. Broadcast the points of the trial surface to all cells. If a cell contains trial surface points, interpolate evolution variables to those points.
3. Send the interpolated evolution variables on the trial surface points to a single process.
4. Perform one iteration of the fast-flow algorithm. Get a new trial surface (or finish if the algorithm converges and an apparent horizon is found).
5. Go to step 2.

The difficulty of implementing a horizon finder with task-based parallelism arises in steps 1 and 2 above. As mentioned before in the I/O example (§ 2.2.2), cells are free to evolve as far into the future as they can regardless of what other cells may be doing (so long as a cell has sufficient data from neighboring cells). This poses a significant problem because the iterative horizon finding algorithm requires the evolution variables to persist at a given time long enough for all iterations to finish and for the algorithm to converge. We could easily solve this like we did for the synchronous case where all cells stop at a given time until the horizon-find is finished. However, this defeats the advantages of asynchronous parallelism. The main quandary is then how to store the evolution variables from cells while also allowing those cells to evolve past the time a horizon-find is supposed to happen. A secondary concern is what to do if a cell evolves past *multiple* horizon-find times.

We implement two different solutions. The first is having a passive thread, called the horizon interpolator, whose entire job is to receive volume evolution variables from cells, store and sort them by time, receive a trial surface at a given time, and determine if enough cells have sent their evolution variables at that time to complete the interpolation. If so, then the interpolation is done, and one iteration of the fast-flow algorithm is performed. Once an iteration is finished and a new trial surface is found, the horizon interpolator is once again asked to interpolate onto the new trial surface.

However, we must be careful now because this new trial surface is different from the previous one, and thus may have moved between cells, as shown in Fig. 2.1. Therefore, the horizon interpolator must now check again if it has received enough data from the cells so that the interpolation onto the new trial surface can be completed. If not, the horizon interpolator must wait for data from additional cells. It can sometimes be the case that enough cells have sent their evolution variables

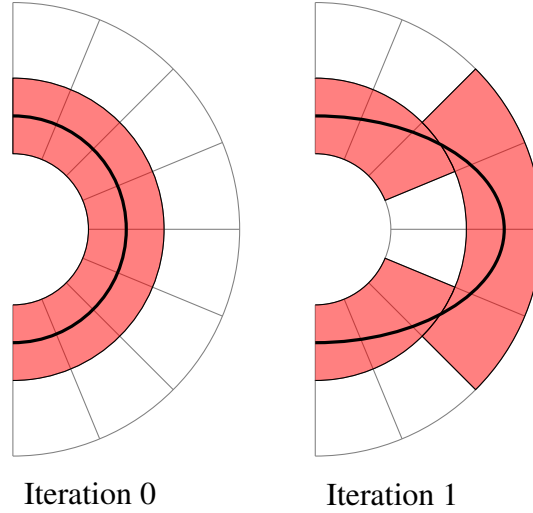


Figure 2.1: Diagram of how trial horizon-find surfaces can move between cells. The thick black lines are the trial surfaces. Cells containing the trial surface are shaded red. *Left*: Trial surface for fast-flow iteration 0 that passes through all cells in the inner ring. *Right*: Trial surface for fast-flow iteration 1 where the surface has moved to change which cells are intersected.

for the first iteration to finish (left panel of Fig. 2.1), but not for the second iteration (right panel of Fig. 2.1); i.e., if only the inner ring of cells in the figure have sent data before it is time to do Iteration 1. This implies our asynchronous algorithm must be able to dynamically receive data from new cells and store them.

There is also a scenario that often happens where a horizon-find will have finished, but the horizon interpolator will still be receiving evolution variables from cells that are slower in their evolution but were not needed to interpolate to any trial surfaces. Therefore, our asynchronous algorithm must be smart enough to receive these variables from late cells, determine that they are no longer needed, and dispose of them to avoid a sort of memory leak similar to the one described in § 2.2.3. This necessitates keeping ledgers of horizon-find times that are currently being worked on, that are waiting to be worked on because a previous horizon-find hasn't finished, and that have already finished.

Another possible scenario that must be taken into account involves the very first horizon-find of the simulation. Consider the situation where a cell has advanced far enough in time so that it is able to send its evolution variables to the horizon interpolator at two different times; an earlier time and a later time. Because of Statement 1, it is possible that the first message the horizon interpolator receives is the later time. Upon initial consideration, this doesn't seem to affect things too

much. This could be fixed by having the cell send the time along with the evolution variables, and then the horizon interpolator could sort its received data by time. However, now consider the entirely possible scenario that the evolution variables at the later time for *all* cells arrive at the horizon interpolator before any of those at the earlier time. The horizon interpolator must somehow know to *not* start a horizon-find at the later time until the earlier time finishes, because it should use the result at the earlier time to form its initial trial surface at the later time. Making the horizon interpolator aware of the initial time unfortunately won't work because horizon-find times are determined dynamically. So it is not known ahead of time when the first horizon-find will occur. Our solution to this problem is to send to the horizon interpolator the time at which the previous horizon-find happened along with the time of the current horizon-find and the evolution variables. This works because a cell knows when the previous horizon-find happened. For the first horizon-find time, the cell sends a sentinel value (e.g. NaN) for the previous horizon-find time, which the horizon interpolator is programmed to recognize as indicating there's no previous time. This allows the horizon interpolator to act accordingly in the scenario described in this paragraph and complete the horizon-finds in order.

Our second interpolation solution is to have the horizon interpolator be part of the horizon finding thread. Almost all of the details between these two approaches are the same, except that when the interpolator is part of the horizon finder the "communication" between the two is just reading and writing local variables. However, the concerns regarding deadlocks and memory management remain since we still want to start the horizon find as soon as all necessary data is available. There are two major differences, though. 1) In the second solution, we only send data from elements that either contain the previous horizon or whose nearest neighbor contains the previous horizon. 2) In the second solution, we communicate the variables needed for the horizon finder instead of the evolved variables, moving some of the computational load from the horizon finding thread to the elements for better parallelization. Because of the similarity in implementation, we will always refer to the "horizon interpolator" to distinguish the part of horizon finding algorithm that handles receiving volume data from the evolution to the trial surface from the horizon finder itself.

The effect of asynchronous implementations is that the process of actually finding the horizon is largely decoupled from the evolution. Cells simply send their evolved variables on their grid points to the horizon interpolator and continue on evolving.

In the synchronous implementation, by contrast, cells would have to wait for a horizon-find to finish before evolving forward again. Moreover, in the evolution of a binary black hole system, there are two (and near merger, three) horizon finds that need to happen, so in the synchronous implementation cells would have to wait twice as long. However, in the asynchronous implementation there can just be two horizon interpolators, one for each horizon, and the only extra cost incurred by the cells is having to send one additional message containing their evolved variables to a second process.

The drawbacks with the asynchronous implementation are the code complexity required to account for the numerous different scenarios that can occur when messages are received and tasks are run in an effectively random order, and the necessity of retaining copies of the volume data for use by the interpolator after the cells have evolved past each horizon-find time.

2.4 Synchronous Control System

We next move on to explaining how the horizon finder communicates back to the black hole evolutions using a feedback control system. We start by explaining the synchronous aspects of this algorithm as there are a number of them that are separate from the asynchronous aspects. The synchronous parts are largely the same as described in refs. [38] and [39], and we summarize them here to point out a few subtle differences that are necessary for an asynchronous implementation. We detail the asynchronous parts in § 2.5.

SpECTRE evolves the first-order Generalized Harmonic (FOGH) formulation of Einstein's equations [40] using the Damped Harmonic (DH) gauge condition [41]. In order to handle a singularity, SpECTRE excises a portion of the computational domain around each singularity. However, this excision introduces a new external boundary of the computational domain. Normally, this would require that we provide some sort of boundary condition for this new boundary. However, we place the boundary inside the apparent horizon of a black hole, and we appropriately control the relative velocity of this boundary with respect to the apparent horizon so that all characteristic fields of the system at this boundary are flowing out of the computational domain and into the black hole. In this case, mathematically a boundary condition is *not* required there, so we do not impose one. However, if the position or velocity of the excision boundary changes in a way such that the characteristic fields are no longer all outgoing, the simulation will fail because

of the missing boundary condition. In order to avoid this, we employ a feedback control system to dynamically adjust the position, shape, and velocity of the excision boundary using time-dependent coordinate mappings to keep the excision within the apparent horizon and keep the characteristic fields outgoing (into the hole).

2.4.1 Computational Domain

Since the problem we'd like to parallelize is the computation of the FOGH equations, our domain decomposition is necessarily intimately tied to how we distribute the work among our computational resources. Asynchronous parallelism is best utilized with many small tasks that can run quickly. A “small” task here means the work done to compute the time step for a number of grid points in the computational domain is small compared to the work to take a time step for all the grid points. The benefit of asynchronous parallelism is wasted if you have a small number of large tasks because then you can only utilize a limited number of computational resources; i.e., the problem won't scale well. Alternatively, if a task is *too* small, then the overhead of sending/receiving asynchronous messages can dominate the cost of the simulation. Therefore, a balance must be struck.

To ensure we have many adequately small tasks, SpECTRE's computational domain is broken up into partitions called *blocks*. A block can be refined into numerous cells called *elements*. Each element contains grid points that are stored contiguously in memory. The job of an element is to then time integrate the FOGH equations at its grid points. This job is accomplished by running numerous tasks on the element. An example of a task could be an element receiving and storing flux information from a neighboring element, then attempting to integrate the FOGH equations one step further in time. The bulk of the tasks run on an element should, in some way, attempt to time integrate the FOGH equations. While an element is executing a task, it can also launch tasks such as asynchronous I/O 2.2.2 or horizon finding 2.3 on other processors.

Fig. 2.2 shows examples of computational domains with excisions in SpECTRE used for single black hole evolutions. The thick black lines are element boundaries and the thin grey lines are grid points. The blue shaded regions are the equatorial projections of a single block. The coordinates for these grid points in Fig. 2.2 are considered the grid coordinates, defined in § 2.4.2. The central excision surface must be kept within an apparent horizon for the entire evolution.

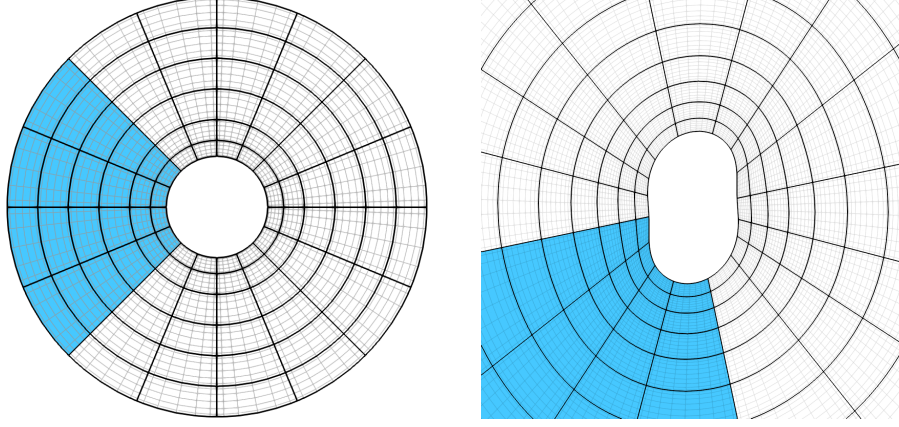


Figure 2.2: Equatorial slices of 3D excised spherical computational domains used for the evolution of single black holes in SpECTRE [42]. The thicker black lines represent element boundaries, and the thinner grey lines represent grid points. The blue shaded regions represent blocks of SpECTRE’s domain. *Left:* An undistorted spherical domain. *Right:* A spherical domain with time dependent coordinate mappings distorting and rotating the excision.

2.4.2 Time-Dependent Coordinate Mapping

In order to keep an excision surface inside an apparent horizon throughout an evolution, we implement time-dependent coordinate mappings that move and distort the computational domain. An example of a domain after time-dependent coordinate mappings have been applied is shown in the right panel of Fig. 2.2. Our implementation closely follows that of [38] and [39] except for a few subtle (yet critical) differences to allow for use with asynchronous parallelism. Here we give enough of a summary of refs. [38] and [39] so that we can explain the necessary changes.

We start with a computational mesh that is described by coordinates $\{\hat{t}, \xi^i\}$ which we will call the “grid coordinates”. These grid coordinates are stationary and do not change over the course of an evolution. They can be thought of as a sort of comoving coordinates. An example of grid coordinates for a single black hole is shown in the left panel of Fig. 2.2. We then apply a coordinate mapping of the form

$$t = \hat{t} \tag{2.1}$$

$$x^i = \mathcal{M}(\lambda(t))\xi^i \tag{2.2}$$

where $\mathcal{M}(\hat{t})$ is a (generally) non-linear operator acting on the grid coordinates. We call $\{t, x^i\}$ the “inertial coordinates” as they represent coordinates where a binary system would orbit and have the separation shrink. An example of inertial coordinates for a single black hole is shown in the right panel of Fig. 2.2. From now

on, we will only use t , not $\hat{t}$, when referring to the coordinate time since they are the same.

The spatial mapping $\mathcal{M}(t)$ is a composition of analytic functions applied to the grid coordinates ξ^i . For example, a 2D rotational mapping $\mathcal{M}_{R2D}$ could take the form

$$\mathcal{M}_{R2D} : \begin{pmatrix} x^1 \\ x^2 \end{pmatrix} = \begin{pmatrix} \cos \theta(t) & -\sin \theta(t) \\ \sin \theta(t) & \cos \theta(t) \end{pmatrix} \begin{pmatrix} \xi^1 \\ \xi^2 \end{pmatrix}, \quad (2.3)$$

where $\theta(t)$ is a time-dependent function that evaluates to the rotation angle at coordinate time t . Each block contains its own mapping $\mathcal{M}_B(t)$, with the only requirement that neighboring blocks' mappings be continuous at the boundaries. They need not be differentiable at block boundaries. Certain mappings can cover the entire computational domain, i.e. a global mapping; or they can cover only part of the computational domain, i.e. a quasi-local mapping. We use the term “quasi-local” and not “local” because each element shares the same mapping as its parent block. Therefore, even if a mapping is in only one block of the computational domain, it is likely shared across multiple elements, and therefore across computational resources. Also, while a mapping may only be quasi-local and not completely global in the topological sense, it can be thought of as global in the computational sense since it can span multiple processors and often times multiple nodes.

We can generalize this to any mapping $\mathcal{M}(t) \equiv \mathcal{M}(\lambda(t))$ where $\lambda(t)$ is a general time-dependent function. We cannot choose $\lambda(t)$ to be some known analytic function because we do not know the trajectory or shape of the apparent horizon for the entire inspiral and merger. The trajectory and shape of the apparent horizons are determined by the solution of Einstein's equations we are trying to obtain. Even post-Newtonian approximations [43] for the trajectories are not accurate enough for our needs since the error in the trajectory would accrue throughout the inspiral.

Instead we choose to represent $\lambda(t)$ as a piecewise polynomial of the form

$$\lambda(t) = \sum_{n=0}^N \frac{1}{n!} (t - t_i)^n \lambda_i^{(n)} \quad \forall \quad t_i < t \leq t_{i+1} \quad (2.4)$$

with N being the degree of each piece of the polynomial and $\lambda_i^{(n)}$ are constants within $t_i < t \leq t_{i+1}$. The reason for this choice of functional form for $\lambda(t)$ is explained in § 2.4.3. Note that we put (n) in parentheses in $\lambda_i^{(n)}$ to signify that it is not an exponent, but just a label. At t_{i+1} we choose a new set of constants λ_{i+1} such

that

$$\lambda_{i+1}^{(N)} = \frac{d^N \lambda}{dt^N} = U(t_{i+1}) \quad (2.5)$$

$$\lambda_{i+1}^{(n)} = \left. \frac{d^n \lambda}{dt^n} \right|_{t=t_{i+1}} \quad \forall \quad 0 \leq n < N \quad (2.6)$$

where $U(t_{i+1})$ will be defined in § 2.4.3. This ensures that all derivatives of λ are continuous except the N th derivative.

There is now an ambiguity of which set of coefficients to use at time t_{i+1} , i.e. $\lambda_i^{(n)}$ vs $\lambda_{i+1}^{(n)}$. We choose to use the coefficients $\lambda_i^{(n)}$ at t_{i+1} to ensure a consistent evolution. Note that during the simulation, we write the values of all t_i and $\lambda_i^{(n)}$ to disk, so that any postprocessing application that uses the output of the simulation will be able to reconstruct the inertial coordinates at any arbitrary time. When postprocessing a simulation, we want the coordinate mappings to return the exact same values at a given time as they did during the evolution. If we used the $\lambda_i^{(n)}$ coefficients at t_{i+1} during the evolution (because the new ones hadn't been computed yet), but during the postprocessing we used the $\lambda_{i+1}^{(n)}$ coefficients (because we now had them), this would produce different values for the coordinate mappings at the same time. Our postprocessed data would then be inconsistent with the data from the evolution. Additionally, choosing $\lambda_{i+1}^{(n)}$ is necessary for an asynchronous implementation (see § 2.5).

2.4.3 Control Error and Signal

For the time-dependent coordinate mappings $\mathcal{M}(\lambda(t))$ described in § 2.4.2, we choose the form of $\lambda(t)$ in Eq. (2.4) because we *a priori* cannot predict the motion and shape of the excision required to stay inside the apparent horizon. Therefore, $\lambda(t)$ needs to be dynamically updated as the simulation progresses based on the evolution of the apparent horizons. We achieve this by applying control theory to the time-dependent coordinate mappings in order to dynamically update the N th derivative of $\lambda(t)$.

We follow the definitions described in §3.1 of [39]. Define a control error Q to be a measure of how incorrect a map parameter $\lambda(t)$ is. If we can define a target value λ_{target} that does not depend on $\lambda(t)$, then we define

$$Q = \lambda_{\text{target}} - \lambda(t). \quad (2.7)$$

If we cannot define λ_{target} because our system is nonlinear and the target value would

depend on $\lambda(t)$, we require that

$$\frac{\partial Q}{\partial \lambda} = -1 + \mathcal{O}(Q) \quad (2.8)$$

In both cases we want to drive $Q \rightarrow 0$ so that $\lambda(t)$ (i.e. the position and shape of the excision boundary) is driven towards λ_{target} (i.e. position and shape of the apparent horizons).

Next, we define the control signal $U(t)$ which we will use to reset the highest derivative of the polynomial representation of $\lambda(t)$ in Eq. (2.5). We use a proportional N th derivative (PND) controller of the form

$$U(t) = \sum_{k=0}^K a_k \frac{d^k Q}{dt^k}, \quad (2.9)$$

where a_k are constants that are chosen to critically damp Q to 0 on a timescale τ_d . Typically $K = 2$. The reason we use a PND controller instead of the typical proportional-integral-derivative (PID) controller used in numerous other applications of control theory is from years of experience using a PND controller in SpEC [39].

In order for this scheme to remain stable, we demand that $\tau_d > t_{i+1} - t_i$. This allows us to assume that $Q(t)$ and $\lambda(t)$ are approximately smooth over the timescale τ_d . With this assumption we are now able to compute the time derivatives of $Q(t)$ and the coefficients a_k necessary for computing $U(t)$ in Eq. (2.9). It is shown in ref. [39] that $U(t)$ is given by

$$U(t) = \frac{d^N \lambda}{dt^N} = -\frac{d^N Q}{dt^N}, \quad (2.10)$$

which can be substituted into Eq. (2.9) and then solved to get $Q \propto e^{-t/\tau_d}$. To get a critically damped solution for a typical $K = 2, N = 3$ PND system, we choose $a_0 = 1/\tau_d^3$, $a_1 = 3/\tau_d^2$, and $a_2 = 3/\tau_d$.

In order to compute the time derivatives of $Q(t)$, we measure and record $Q(t)$ for M times between each t_i and then use Lagrange interpolation polynomials to get $d^k Q/dt^k$. Typically we use $M = N + 1$ and use K th order non-uniformly spaced stencils for the derivatives. For parameters that have small τ_d , the measurements are often done once per time step. For parameters with larger τ_d , measurements can be done less frequently. Since τ_d can dynamically change over the course of a simulation, how often we measure $Q(t)$ will change as well.

There are additional procedures outlined in §3 of [39] related to averaging out noisy measurements of $Q(t)$ and also dynamically adjusting the damping timescale τ_d of

a control system. These are also implemented in SpECTRE, but neither of these affect the aspects of the control system and horizon finding related to synchronous parallelism. Accordingly, we refer the reader to [39] for details. Dynamically adjusting the damping timescales can affect the asynchronous parallelism and will be discussed in § 2.5.

2.4.4 Computing $\lambda^{\text{new}}(t)$

Now that we have gone through the necessary pieces of horizon finding (§ 2.3), the computational domain (§ 2.4.1), the time-dependent coordinate mappings (§ 2.4.2), and the control signal (§ 2.4.3), we detail the process that occurs in a synchronous application to dynamically update the time-dependent coordinate mappings. For simplicity, we will assume the following: there is only one time-dependent map parameter $\lambda(t)$ that we are controlling; the order of $\lambda(t)$ is $N = 3$; the order of the PND controller is $K = 2$; we measure $Q(t)$ $M = 4$ times; and τ_d is small enough that we measure $Q(t)$ every time step.

In the previous section, we discuss and treat $\lambda(t)$ as a function that spans the entire evolution that we periodically update. We will continue to do so, but we also define two new quantities, $\lambda^{\text{new}}(t)$ and $\lambda^{\text{old}}(t)$, for convenience. $\lambda^{\text{new}}(t)$ refers to $\lambda(t)$ once the new coefficients λ_{i+1} have been computed and used. $\lambda^{\text{old}}(t)$, then refers to $\lambda(t)$ before these new coefficients have been computed.

The (synchronous) steps to compute $\lambda^{\text{new}}(t)$ are:

1. Wait for all elements to finish their time step. Evolution variables are now in a consistent state at the same time.
2. Find the apparent horizon (§ 2.3).
3. Calculate and store $Q(t)$ using the apparent horizon
4. If not the $M = 4$ th measurement, go to step 1
5. At the $M = 4$ th measurement, compute $U(t)$ using $d^k Q/dt^k$ and compute a new τ_d .
6. Use $U(t)$ to set $d^N \lambda/dt^N$ at t_{i+1} .
7. Use τ_d^{new} to possibly set the new time step because we require that $\tau_d > t_{i+1} - t_i$. Broadcast $d^N \lambda/dt^N$ and new time step to all nodes/cores.

8. On each node that receives $d^N \lambda / dt^N$, use Eqs. (2.5)-(2.6) and $\lambda^{\text{old}}(t)$ to compute $\lambda^{\text{new}}(t)$.
9. Go to step 1

With this procedure, all elements must wait until the horizon-find finishes and until all control system quantities are computed and broadcast before continuing the evolution. The advantage of the synchronous implementation is that everything happens deterministically and in order, making the logic-flow of the code easy to follow. The disadvantage is that during the $M = 1, 2, 3$ measurements of $Q(t)$, which include horizon-finds, no global communication of time-dependent parameters needs to happen. This implies the elements could, in theory, keep evolving until the $M = 4$ measurement without stopping and waiting (assuming the horizon finder is implemented asynchronously as in § 2.3). This is precisely the performance improvement we employ with our asynchronous approach in § 2.5.

2.5 Asynchronous Control System

Here we detail the differences from the previous section when implementing an asynchronous feedback control system that mutates a global quantity in a simulation; i.e., dynamically updating the time-dependent map parameters in SpECTRE.

2.5.1 Global State

In § 2.4.2, we briefly mentioned that the time-dependent coordinate mappings can be quasi-local in the sense that they can cover multiple elements. In turn, because elements are then placed on many cores/nodes, this means the mappings can also span multiple cores/nodes. Therefore the mappings need to be kept consistent across the computational resources in order to have a consistent evolution. This implies that the time-dependent coordinate mappings (or more accurately the time-dependent map parameters $\lambda(t)$) are what we call a “global state” of the simulation. Such a global state wouldn’t be an issue if it were constant or immutable, because then it could just be set at the beginning and would never change during the simulation. However, $\lambda(t)$ *does* need to be mutated using the feedback control system. Therefore, we don’t just have a global state, but a mutable global state. A mutable global state can cause issues in an asynchronous application because keeping the state of the simulation consistent can be quite difficult and error prone.

One of the common wisdoms when dealing with any asynchronous application is to avoid a mutable global state whenever possible. However, in our case we cannot

avoid such a global state because of the nature of our quasi-local time-dependent coordinate mappings. One common way of dealing with a mutable global state when porting a synchronous application to be asynchronous is called privatization of the global state [44]. Instead of the global state being located in a common place accessible to all tasks, a local copy of the global state is stored in each task. Then, a task simply continues until its local copy of the global state is invalid and/or it receives a new global state.

While this is theoretically possible in our case, privatization becomes impractical with a large number of tasks and with a global state that grows in size over time. Our production-level simulations will typically have $10^3 - 10^5$ elements (tasks) and the size of $\lambda(t)$ (as parameterized by the coefficients $\lambda_i^{(n)}$) will grow to around 75 MB by the end of the simulations (because of how many times we mutate $\lambda(t)$ and the necessity to store all past history described at the end of § 2.4.2). If we were to store $\lambda(t)$ locally on each task, this would require 75 GB – 7.5 TB of total memory. Even for modern high-performance computing systems, this is likely too much. Therefore we keep only a single copy of $\lambda(t)$ in a shared place on a node, mutate that copy, and algorithmically ensure access to this shared mutable global state is consistent. An alternative solution which we do not employ would be to use privatization of $\lambda(t)$ but only store recent history in memory, and periodically write older history to disk and purge it from memory.

2.5.2 Expiration Time

Before we detail how we mutate $\lambda(t)$ consistently, we must explain one of the most critical (but also subtle) differences between the synchronous and asynchronous control system implementations, which comes from t_{i+1} in Eq. (2.4). In the synchronous case, because the evolution was paused at t_{i+1} in order to find the horizon and then compute $\lambda^{\text{new}}(t)$, no elements will evolve past t_{i+1} and try to evaluate $\lambda(t)$ at a time $t > t_{i+1}$. Consistency in using $\lambda(t)$ is thus built into the synchronous algorithm.

However, in the asynchronous case this is now an issue. Finding the apparent horizon is an asynchronous algorithm, so the elements won't pause to wait for a horizon-find to finish. Therefore, so long as they are able, the elements will keep evolving forward in time and eventually past t_{i+1} . This can cause an inconsistent state in the simulation. Take the case where an element will evolve past t_{i+1} , the $M = 4$ measurement of the horizon occurs, and $\lambda^{\text{new}}(t)$ is computed and broadcast.

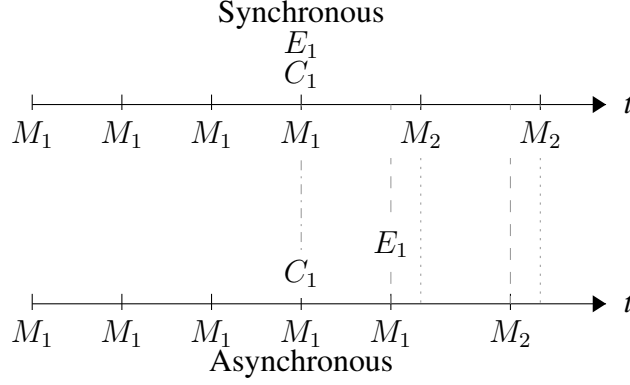


Figure 2.3: Timeline for computing $\lambda^{\text{new}}(t)$ for both the synchronous (*top*) and asynchronous (*bottom*) case. Here M_j denotes when we perform a measurement (horizon-find), C_j denotes when we compute $\lambda^{\text{new}}(t)$, and E_j denotes the expiration time which is when we mutate $\lambda^{\text{old}}(t) \rightarrow \lambda^{\text{new}}(t)$ and set a new expiration time. In the synchronous case, we compute and mutate $\lambda(t)$ at the same time because all elements are already paused. In the asynchronous case, we delay mutating $\lambda(t)$ by one (old) measurement for better utilization. The vertical dashed and dotted lines show how measurements (fail to) line up between the synchronous vs. asynchronous cases.

If $\lambda^{\text{new}}(t)$ arrives at the node the element is on *before* the element evolves past t_{i+1} , then the element will use $\lambda^{\text{new}}(t)$ for its evolution. However, if the $\lambda^{\text{new}}(t)$ arrives *after* the element evolves past t_{i+1} , then the element will have used the outdated $\lambda^{\text{old}}(t)$ for its evolution and will be incorrect. Furthermore, with many elements in a simulation on different nodes, it's possible that some elements use the correct $\lambda^{\text{new}}(t)$ and some use the incorrect $\lambda^{\text{old}}(t)$ because of Statement 1 and message ordering, resulting in a completely inconsistent evolution.

To address this potential inconsistency, we conceptually promote t_{i+1} from being the end of the interval of $\lambda(t)$ to what we call an “expiration time”. With this promotion, we require that elements cannot evolve past an expiration time. Instead, if they otherwise would be able to evolve past the expiration time, they must pause and wait to receive $\lambda^{\text{new}}(t)$ before continuing. In practice we have many different $\lambda(t)$ for the various time-dependent coordinate mappings SpECTRE uses, so an element must obey the expiration time of each $\lambda(t)$ (i.e., the $\min\{t_{i+1}^j\}$). In computer science language, we use t_{i+1} to establish a *happens-before* relation between tasks.

Even though we have fixed the consistency issue, we need to also consider efficiency and utilization when obeying the expiration time. For the synchronous case, we perform a horizon-find at the same time as we compute and mutate $\lambda^{\text{new}}(t)$. Again,

for the synchronous case this is irrelevant because the evolution of the elements is already paused. However, when we consider the asynchronous case, if we keep the same structure of requiring a horizon-find at the expiration time, the elements will send their data to the asynchronous horizon finder and then immediately have to wait for $\lambda^{\text{new}}(t)$. This is not efficient, since a horizon-find will always have to happen at the expiration time and elements will always have to wait, causing low utilization of the computational resources. A different mutation structure is needed in order to minimize how much time the elements wait.

This new mutation structure is depicted in Fig. 2.3. The horizontal axes represent time. The top axis is for the synchronous case, and the bottom axis is for the asynchronous case. Here M_j denotes when we perform a measurement (horizon-find), C_j denotes when we compute $\lambda^{\text{new}}(t)$, and E_j denotes the expiration time t_{i+1} when we actually mutate $\lambda^{\text{old}}(t) \rightarrow \lambda^{\text{new}}(t)$. For the synchronous case, $C = E = t_{i+1}$ and the first four measurements are equally spaced at Δt_{M_1} while the subsequent two are also equally spaced but at $\Delta t_{M_2} > \Delta t_{M_1}$ to denote a change in τ_d .

For the asynchronous case, when we start a measurement that will eventually compute $\lambda^{\text{new}}(t)$, we don't want the elements to stop immediately. We'd like to have them continue evolving for a short while so the horizon-find has a chance to finish and we can mutate $\lambda^{\text{old}}(t) \rightarrow \lambda^{\text{new}}(t)$ before the element reaches the expiration time. Therefore we need to decouple C and E in Fig. 2.3 for the asynchronous case. We achieve this by delaying E by one extra measurement of $\lambda^{\text{old}}(t)$. Then, when we start the measurement C that will be used to compute $\lambda^{\text{new}}(t)$, the elements still have time left before they hit the expiration time. They can evolve, allowing forward evolution progress to happen while the horizon-find for C happens.

The choice of how long to delay E is somewhat arbitrary. If it's too far into the future, outdated horizon information will be used to compute $\lambda^{\text{new}}(t)$; too soon after C and the elements will be quickly blocked by the expiration time. We found one (old) measurement time to be a suitable balance between these two concerns. Additionally, this choice fits nicely into the existing implementation infrastructure.

We note that this change of delaying the mutation by one (old) measurement time means that we will be using "outdated" information when we mutate $\lambda(t)$ at E . In the bottom axis of Fig. 2.3 this is shown with the measurement M_1 which occurs at the same time as E_1 but will be used as the first measurement for computing C_2 (not shown). This measurement M_1 is performed with $\lambda^{\text{old}}(t)$, not $\lambda^{\text{new}}(t)$, and therefore

uses “outdated” information. An improvement could be to extrapolate the value computed at C to the time of E . Then, this extrapolated value C^{extrap} will act as if we used the measurement M at E to compute C (i.e. them all occurring at the same time like the synchronous case) while still maintaining the asynchronous advantage of delaying the mutation of $\lambda(t)$ by one (old) measurement.

2.5.3 Querying $\lambda(t)$ and Computing $\lambda^{\text{new}}(t)$

Now that we have an efficient prescription for when to compute $\lambda^{\text{new}}(t)$ and mutate it, the last piece we need is a method to actually query $\lambda(t)$ and then perform the mutate in a consistent way. As a reminder, $\lambda(t)$ is a shared mutable global state, meaning that many different elements will all access the same copy of $\lambda(t)$ in memory on a node. Additionally, all $\lambda(t)$ are stored in a single data structure we’ll denote as $\Lambda(t) = \{\lambda_k(t)\}$. This is because we actually have several time-dependent coordinate mappings, each using one or multiple of the $\lambda_k(t)$. Therefore, care needs to be taken when mutating $\Lambda(t)$ since elements may be using it simultaneously.

The most naive method for ensuring this consistency is to allow only one element/task to query or mutate $\lambda(t)$ (or $\Lambda(t)$) at a time. However, this would be extremely inefficient, especially for a large number of elements on a single node. Therefore, we put two restrictions on how $\lambda(t)$ can be queried and mutated, which will inform what data structure we can use to represent $\Lambda(t)$. We first require that $\Lambda(t)$ cannot be mutated by two different cores simultaneously (i.e. two different $\lambda_k(t)$). While not strictly necessary, this assumption simplifies the code significantly and we have not found this to affect performance. The second requirement is that each $\lambda_k(t)$ must be queryable at all wallclock times, even while it is being mutated. A useful case is for elements that are slower than others. Slower elements may need to query one $\lambda_k(t)$ (and thus don’t depend on a mutate to $\Lambda(t)$) and should be able to do so without waiting for a mutation of a different $\lambda_l(t)$ to finish.

A data structure that satisfies the above requirements is a linked list for each $\lambda_k(t)$ where access to the most recent entry of each list is protected by an atomic variable. Then those lists are stored in a map data structure. Since we store all past history, these linked lists will only be appended to and their entries will never be changed. The thread safety of each list when being queried and mutated simultaneously is guaranteed by the atomic variable, which is more efficient than locks.

Next, we discuss the procedure for how elements can query $\Lambda(t)$, which has some subtleties. A key feature for this procedure is the concept of a “callback”. A callback

Algorithm 1: Registering a callback with $\Lambda(t)$

```

1 if  $t \leq t_{i+1}$  then
2   No callback is registered; value is fetched
3 else
4   Element registers a callback with  $\Lambda(t)$ 
5   if  $t \leq t_{i+1}$  then
6     If a callback exists, remove it; value is fetched
7   else
8     A callback was already registered. Don't register another
9   end
10 end

```

is some function to be run after some other event (function, communication, task) has finished its execution. Callbacks are used in our algorithm when an element checks if it can evaluate $\Lambda(t)$ at a time after the expiration time. When this happens, the element pauses its evolution and “registers” a callback with $\Lambda(t)$. Registration of a callback involves locking the list of existing callbacks for $\Lambda(t)$, appending the incoming callback to the list, then releasing the lock. Once a $\lambda_k(t)$ is mutated, this callback will restart the evolution of the element.

The subtleties arise based on the order of events happening; i.e., whether $\lambda_k(t)$ is being mutated just before or just after the element registers a callback with $\Lambda(t)$. The details are described with pseudocode in Algorithm 1. When an element queries $\Lambda(t)$, it first checks if the time it's querying is after the expiration time. If so, it registers a callback. If not, no callback is registered. Then we check the same condition a second time immediately afterwards. This is because of how we define $\Lambda(t)$. Therefore, suppose that the if-else starting at line 5 in Algorithm 1 was omitted. Then the following events could occur sequentially in wallclock time:

1. An element queries $\Lambda(t)$ for $\lambda_0(t)$ and registers a callback.
2. $\lambda_1(t)$ is mutated, copies and invokes all callbacks of $\Lambda(t)$ restarting the element.
3. The element again queries $\Lambda(t)$ for $\lambda_0(t)$ and finds it still hasn't been mutated, but has not yet registered a callback (between lines 3 and 4 in Algorithm 1).
4. $\lambda_0(t)$ is mutated *after* the element queried for $\lambda_0(t)$, found no callbacks, and does nothing.

5. Now the callback from the query in step 3 is registered, and the element is paused.

We now have a scenario where the element is paused waiting for a change that has already occurred in step 4, and our simulation will deadlock (§ 2.5.6). This is why we must check the same condition a second time after step 5. This second check is line 5 in Algorithm 1. When we do this, we'll conclude that $\lambda_0(t)$ is in fact ready, remove the callback which is now unneeded, and the element can proceed. If $\lambda_0(t)$ still wasn't ready, then we have already registered a callback so the element will be paused and we can be assured that the mutation of $\lambda_0(t)$ will come sometime in future wallclock time.

Finally, we discuss how to mutate $\Lambda(t)$ which is considerably more straightforward than querying $\Lambda(t)$ because we only allow one mutation of $\Lambda(t)$ at a time. To mutate $\Lambda(t)$, we lock $\Lambda(t)$ to prevent other mutations and append the new $\lambda_i^{(n)}$ coefficients (Eqs. (2.4) – (2.6)) to the linked list of the $\lambda_k(t)$ that's being mutated. After we release the mutation lock, we lock the vector of callbacks associated with $\Lambda(t)$, copy them, release the callback lock, and invoke each callback in the copied list. The reason for the copy is because some callbacks may immediately register another callback (as explained in the previous paragraph), so we need to avoid an infinite recursion of callbacks being invoked and then those same callbacks being re-registered.

Additionally, if a callback is registered between the wallclock time when we release the mutation and acquire the callback lock, this is not an issue. Whether $\Lambda(t)$ is ready or not, this newly registered callback will still be called. If $\Lambda(t)$ is ready, then the callback will likely restart the evolution of an element. If a $\lambda_k(t)$ still isn't ready, then a new callback will just be registered when a different $\lambda_l(t)$ in $\Lambda(t)$ is ready.

2.5.4 Eventual Concurrency

Yet another subtlety of dealing with a mutable global state is a concept we'll call eventual concurrency. Concurrency on its own broadly refers to running or executing multiple tasks simultaneously and consistently on some shared resources [45]. Eventual concurrency then refers to scenarios where concurrency will need to be adhered to at some point in the future, but not necessarily right now. This is probably explained most easily through another example.

Consider that we have element A on node 0 where $\Lambda(t)$ is up-to-date at some time

t , and we have element B on node 1 where $\Lambda(t)$ is *not* up-to-date at time t because the mutation of $\Lambda(t)$ hasn't been received on node 1 yet (due to Statement 1). Now suppose that element A sends a message M to element B, starting work on element B that requires $\Lambda(t)$ to be up to date. Again, because of Statement 1, there is *no guarantee* that M arrives at element B after $\Lambda(t)$ has been mutated. It is entirely possible to arrive before $\Lambda(t)$ is mutated on node 1, even though it was sent after $\Lambda(t)$ was mutated on node 0.

Therefore, element B must query $\Lambda(t)$ (§ 2.5.3) on node 1 before it does its work, even though $\Lambda(t)$ was up-to-date on node 0 when element A sent the message. If element B finds $\Lambda(t)$ not up-to-date, then it registers a callback. If element B finds $\Lambda(t)$ is up-to-date, then it can continue. This necessity of having to always query $\Lambda(t)$ when a task starts its execution, regardless of whether $\Lambda(t)$ is up to date on the node of the sender of the task, is what we call eventual concurrency. In practice, $\Lambda(t)$ only needs to be queried if the task actually uses $\Lambda(t)$. If it doesn't, this check for eventual concurrency can be skipped.

2.5.5 Summary

Previous sections went into detail about each aspect of our asynchronous algorithm. Here, we provide a higher-level summary of the workflow for how the time-dependent maps, their parameters, the control system, the horizon-finds, and asynchronous communication all interact and influence each other.

- The excision boundaries and the mesh (§ 2.4.1) are controlled by time-dependent coordinate maps and their parameters $\lambda(t)$ (§ 2.4.2), which must be updated periodically throughout the simulation as the black holes inspiral and merge.
- Each $\lambda(t)$ is represented as a piecewise polynomial in time (Eq. (2.4)).
- Each piece of $\lambda(t)$ will expire (§ 2.5.2) at some time and elements will have to wait for it to be mutated to continue their evolution past this expiration time.
- For these reasons, $\lambda(t)$ occupies a mutable global state in the simulation (§ 2.5.1).
- Elements must query (§ 2.5.3) $\lambda(t)$ at a given time before they can evolve forward and eventually find a horizon.

- If the given time is after the expiration time, the element registers a callback (§ 2.5.3) for when $\lambda(t)$ is mutated.
- We mutate $\lambda(t)$ (§ 2.4.3) with information from four previous apparent horizon-finds.
- Each horizon-find is done asynchronously and requires novel logic to handle the collection of variables from each element (§ 2.3).
- Once four horizon-finds have happened and $\lambda^{\text{new}}(t)$ is computed, $\lambda(t)$ is mutated in a threadsafe and asynchronously consistent manner (§ 2.5.3).
- Any element that registered a callback for when $\Lambda(t)$ was mutated is then restarted.

2.5.6 Deadlocks and Errors

If any of the asynchronous algorithms described above in sections 2.2.2-2.5 is implemented incorrectly, or if there are edge cases that aren't taken into account, a phenomenon called a “deadlock” can occur. In its simplest form, a deadlock occurs when you have a task that is waiting to receive a message but never does. Here, “waiting” does not mean that a task is still allocated to a core blocking another task from running on the core. It means that a task has finished its computation, is no longer running on a core, and is idle until it can be restarted. A common deadlock can happen when there are two tasks A and B, and A is waiting for a message from B while B is waiting for a message from A. Therefore, both tasks halt indefinitely. One scenario of a deadlock was described in § 2.5.3 when querying and mutating $\Lambda(t)$ happen simultaneously.

The consequence of a deadlock is that every element will continue to evolve until it no longer can. Then, once no elements can continue evolving forward in time, the simulation will hang; i.e. quiesce. Fortunately, Charm++ offers quiescence detection and we have built a feature into SpECTRE that uses quiescence detection to then run additional functions that will terminate the simulation cleanly. However, this termination will likely not happen at the time that the user intended. Elements could be waiting to be restarted at different times in an inconsistent state. This can be quite difficult to debug because the simulation is no longer in the state it was when the actual error/deadlock occurred. Additionally, it is also difficult because the simulation looks as though it finished correctly because there was no obvious error such as an FPE or segmentation fault. Therefore we developed a method to debug

deadlocks that outputs the deadlocked state of the simulation to disk, including copious amounts of information on what messages each task is waiting on. This information can be used to identify the offending task that caused the deadlock.

We'll use the horizon interpolator as an example of the information that is output after a deadlock since it couples many different communication patterns. First, we output which elements have sent their volume variables to the horizon interpolator at given times. If a horizon-find did not complete at a certain time, and the horizon interpolator is missing an element's variables, then we know we need to look at why the horizon interpolator never received the variables. This could be because the element never sent a message, the message got lost, or the message did arrive but the logic in the horizon interpolator didn't handle the message properly. We also output which horizon-finds have completed, are currently being worked on, or are waiting to be worked on, which can help us determine how far forward in time some elements have evolved. Additionally, we output the points of the current fast-flow iteration trial surface and which points already have been interpolated to. This can tell us if interpolation failed for a certain trial point or if some elements haven't sent their variables. Furthermore, we also output the nodes/cores that elements sent their variables from, which can be helpful in determining if the problem is with inter-node messages.

Another useful diagnostic for debugging a deadlock is to output $\Lambda(t)$, whether any callbacks were registered, and which elements those callbacks would have restarted. This can usually be used in conjunction with other deadlock information (such as the horizon interpolator) to determine if an element simply couldn't evolve forward because of an expiration time or because it was waiting for other data necessary for its evolution. It can also be used to ensure that an element *was* restarted and thus direct us to other areas to look for the bug.

2.6 Timing Results

For vacuum simulations, SpECTRE usually uses a local time-stepping algorithm for the volume evolution [30]. The LTS algorithm is incompatible with the synchronous, step-based control-system implementation used in our older code SpEC, and, for simplicity, we have chosen to use the fully asynchronous algorithm even when LTS is disabled. We can, however, adjust the delay in the update of λ to switch between the two update schemes shown in Fig. 2.3, while still using the asynchronous infrastructure. The "synchronous" scheme introduces more dependencies between

the tasks in the evolution, and results in elements waiting for the horizon finder more often than the “asynchronous” scheme.

We compared the performance of three different parallelization schemes when simulating an eccentricity-reduced 4 : 1 mass-ratio BBH, each run on a single 192-core node on the mbot cluster at Cornell. For details on the setup for this problem, see [42], where the spatial resolution used here is similar to that paper’s “Lev1”. Using the full asynchronous algorithm described above, $1000M$ (where M is the total mass) before the merger, the evolution ran at $34.5M/\text{hr}$. Modifying the algorithm to use the “synchronous” update method slowed this to $33.0M/\text{hr}$. Finally, we ran the evolution with the full asynchronous control system, but with LTS disabled, which resulted in a speed of $6.33M/\text{hr}$. This shows that one of SpECTRE’s biggest algorithmic strengths is LTS, gaining us roughly a factor of five in speed, and the other parts of our code must be written to be compatible with that. The “asynchronous” update scheme provides a much smaller, but still noticeable, improvement, and is relatively easy to implement once the method is asynchronous in general.

2.7 Application to Binary Neutron Stars

Even though the techniques described in previous sections were developed for use in black hole simulations, many of them can still be applied to simulations of other systems that are done with asynchronous parallelism. Here we describe how they can be applied to simulations of binary neutron stars (BNSs) as they are done in SpECTRE [46].

The surface of a neutron star is a discontinuity in the matter profile and is usually handled with shock capturing methods [47]. If the computational domain is stationary and a binary neutron star (BNS) system is orbiting, a NS surface will pass by a grid point on a timescale of roughly

$$\tau_{\text{stationary}} \propto \frac{\Delta x}{\Omega d/2}, \quad (2.11)$$

where Δx is the spacing between grid points, d is the separation between the NSs, and Ω is the orbital angular velocity. However, if the computational grid is rotating at the same rate Ω as the stars are orbiting, the surface of the star would move between elements much more slowly. Having the computational domain move with the fluid can limit numerical diffusion (e.g. [48]). If we boost into this corotating frame, the only movement of the stars would be radially inward as they inspiral.

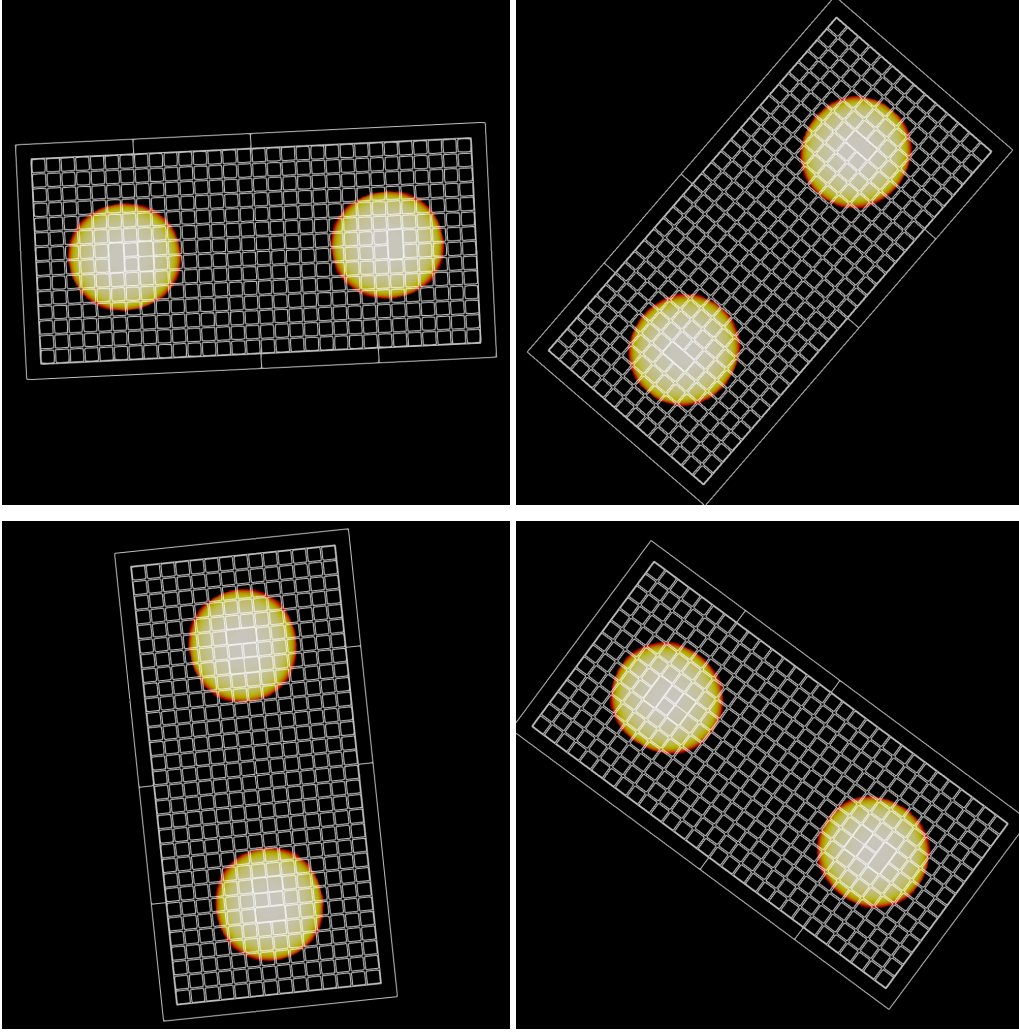


Figure 2.4: Snapshots of the rest mass density and grid of a BNS system where the computational domain is rotated and controlled using the methods in this paper.

They will pass by grid points on a timescale of roughly

$$\tau_{\text{corotating}} \propto \frac{\Delta x}{\dot{r}}, \quad (2.12)$$

where $\dot{r}$ is the radial velocity of the NSs. If we use $\Delta x = 200$ m, $d = 32.4$ km, $\Omega = 1.1 \times 10^3$ rad/s, and $\dot{r} = 11$ km/s (taken from [49]), we have

$$\tau_{\text{stationary}} \approx 0.01 \text{ ms} \ll \tau_{\text{corotating}} \approx 20 \text{ ms}, \quad (2.13)$$

showing how much more slowly the discontinuity of the NS would move through a corotating grid.

To rotate the computational domain with the orbit of the NSs, though, we don't *a priori* know the evolution of the orbital velocity of the BNSs, just like we also cannot

know it for BBHs. Therefore, we need a dynamic way of tracking the evolution of the BNS system. The overall method in § 2.5 can be applied here to dynamically track and update the angular velocity of the computational domain, except we don't use horizon-finds as our measurements. Instead, we find the center of mass of each NS and the system as a whole and use these measurements to compute $Q(t)$, its derivatives, $U(t)$, and eventually $\lambda(t)$. In Fig. 2.4, we show snapshots of an asynchronous BNS simulation done with SpECTRE where the NSs are tracked with the methods in this paper [49].

2.8 Conclusion

In this paper, we present novel methods for finding and tracking apparent horizons in asynchronous simulations of black holes using a feedback control system. We first introduce asynchronous parallelism in § 2.2.1 and provide a Statement 1 about asynchronous parallelism that is the cornerstone of all our algorithms. Additionally, we present an example of how a common algorithm like I/O must be rethought when implementing it in an asynchronous application compared to a synchronous one. We then detail how our asynchronous horizon finding algorithm works compared to a synchronous implementation in § 2.3. Maybe surprisingly, implementing an asynchronous horizon finder itself is not the most complicated parallel aspect for our dynamic tracking of apparent horizons.

We then present details for a synchronous feedback control system in § 2.4 and its asynchronous counterpart in § 2.5. This feedback control system dynamically adjusts the time-dependent coordinate mappings used to move and distort the computational domain so the position and shape of the excision boundaries match those of the apparent horizons. The critical conclusion of this work is that the time-dependent coordinate mappings act as shared mutable global states in the asynchronous simulation. We describe the extreme care that needs to be taken when dealing with these shared mutable global states to avoid deadlocks, errors, and inconsistencies in an evolution. All these algorithms have been added to the open-source numerical relativity code SpECTRE and have already been used in a number of publications [42, 46, 49–52].

Additionally, in § 2.7 we show that, omitting horizon finding, the asynchronous feedback control system can be used for evolutions of BNSs to rotate the computational domain to follow the orbit of the NSs. This shows the versatility of these algorithms and their potential to be applied to other evolutions where dynamic track-

ing of a feature would be useful. As future work, we plan to apply the methods in this paper to other problems, such as black hole-neutron star binaries or utilizing SpECTRE's flexible domains to track features in other simulations of general relativistic magnetohydrodynamic systems.

Finally, in § 2.6 we study the runtime of a mass ratio 4 non-spinning BBH simulation. The asynchronous horizon tracking algorithm enables the use of local time stepping, which ultimately provides a factor of five decrease in runtime. The cost savings is expected to increase for higher mass ratio and spinning BBH simulations. The asynchronous update algorithm provides a further improvement, though only on the order of a few percent. We believe that asynchronous parallelism and the methods described here will be a critical ingredient in high-performance BBH simulations.

Acknowledgments

KCN would like to thank Geoffrey Lovelace for running SpECTRE BBH simulations and finding various deadlocks that needed to be fixed and accounted for in the algorithms presented in this work. This material is based upon work supported by the National Science Foundation under Grants No. PHY-2309211, No. PHY-2309231, and No. OAC-2209656 at Caltech; by No. PHY-2407742, No. PHY-2207342, and No. OAC-2209655 at Cornell; and by No. PHY-2208014, No. PHY-1606522 and AST-2219109 at Cal State Fullerton. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was supported by the Sherman Fairchild Foundation at Caltech and Cornell, and by Nicholas and Lee Begovich and the Dan Black Family Trust at Cal State Fullerton. Charm++/Converse [33] was developed by the Parallel Programming Laboratory in the Department of Computer Science at the University of Illinois at Urbana-Champaign. The figures in this article were produced with TikZ [53] and ParaView [54, 55]. Computations were performed with the Wheeler cluster and the Resnick High Performance Computing Center at Caltech, the mbot cluster at Cornell, and the Ocean cluster at Cal State Fullerton.

Bibliography

- [1] Junaid Aasi et al. Advanced LIGO. *Class. Quant. Grav.*, 32:074001, 2015. doi: 10.1088/0264-9381/32/7/074001.
- [2] Fausto Acernese et al. Advanced Virgo: A second-generation interferometric

- gravitational wave detector. *Class. Quant. Grav.*, 32(2):024001, 2015. doi: 10.1088/0264-9381/32/2/024001.
- [3] Tomotada Akutsu et al. Overview of KAGRA: Detector design and construction history. *Progress of Theoretical and Experimental Physics*, 2021(5): 05A101, May 2021. doi: 10.1093/ptep/ptaa125.
- [4] Katherine L. Dooley et al. GEO 600 and the GEO-HF upgrade program: Successes and challenges. *Class. Quant. Grav.*, 33:075009, 2016. doi: 10.1088/0264-9381/33/7/075009.
- [5] Benjamin P. Abbott et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, February 2016. doi: 10.1103/PhysRevLett.116.061102. URL <https://link.aps.org/doi/10.1103/PhysRevLett.116.061102>.
- [6] Elenna Capote et al. Advanced LIGO detector performance in the fourth observing run. *Phys. Rev. D*, 111(6):062002, 2025. doi: 10.1103/PhysRevD.111.062002.
- [7] Abdul H. Mroue et al. Catalog of 174 binary black hole simulations for gravitational wave astronomy. *Phys. Rev. Lett.*, 111(24):241104, 2013. doi: 10.1103/PhysRevLett.111.241104.
- [8] Karan Jani, James Healy, James A. Clark, Lionel London, Pablo Laguna, and Deirdre Shoemaker. Georgia Tech catalog of gravitational waveforms. *Class. Quant. Grav.*, 33(20):204001, 2016. doi: 10.1088/0264-9381/33/20/204001.
- [9] James Healy, Carlos O. Lousto, Yosef Zlochower, and Manuela Campanelli. The RIT binary black hole simulations catalog. *Class. Quant. Grav.*, 34(22): 224001, 2017. doi: 10.1088/1361-6382/aa91b1.
- [10] James Healy, Carlos O. Lousto, Jacob Lange, Richard O’Shaughnessy, Yosef Zlochower, and Manuela Campanelli. Second RIT binary black hole simulations catalog and its application to gravitational waves parameter estimation. *Phys. Rev. D*, 100(2):024021, 2019. doi: 10.1103/PhysRevD.100.024021.
- [11] Michael Boyle et al. The SXS collaboration catalog of binary black hole simulations. *Class. Quant. Grav.*, 36(19):195006, 2019. doi: 10.1088/1361-6382/ab34e2.
- [12] Mark A. Scheel et al. The SXS collaboration’s third catalog of binary black hole simulations, May 2025. URL <https://arxiv.org/abs/2505.13378>.
- [13] James Healy and Carlos O. Lousto. Third RIT binary black hole simulations catalog. *Phys. Rev. D*, 102(10):104018, 2020. doi: 10.1103/PhysRevD.102.104018.

- [14] James Healy and Carlos O. Lousto. Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits. *Phys. Rev. D*, 105(12):124010, 2022. doi: 10.1103/PhysRevD.105.124010.
- [15] Deborah Ferguson et al. Second MAYA catalog of binary black hole numerical relativity waveforms. *Phys. Rev. D*, 112(4):044043, 2025. doi: 10.1103/gk7x-9cds.
- [16] Pau others Amaro-Seoane. Laser interferometer space antenna. *arXiv e-prints*, art. arXiv:1702.00786, February 2017. doi: 10.48550/arXiv.1702.00786. URL <https://arxiv.org/abs/1702.00786>.
- [17] Michele Punturo et al. The Einstein Telescope: A third-generation gravitational wave observatory. *Class. Quantum Grav.*, 27(19):194002, October 2010. doi: 10.1088/0264-9381/27/19/194002.
- [18] Matthew Evans et al. A horizon study for Cosmic Explorer: Science, observatories, and community. September 2021. URL <https://arxiv.org/abs/2109.09882>.
- [19] Michael Pürrer and Carl-Johan Haster. Gravitational waveform accuracy requirements for future ground-based detectors. *Phys. Rev. Res.*, 2(2):023151, 2020. doi: 10.1103/PhysRevResearch.2.023151.
- [20] Deborah Ferguson, Karan Jani, Pablo Laguna, and Deirdre Shoemaker. Assessing the readiness of numerical relativity for LISA and 3G detectors. *Phys. Rev. D*, 104(4):044037, 2021. doi: 10.1103/PhysRevD.104.044037.
- [21] Aasim Jan, Deborah Ferguson, Jacob Lange, Deirdre Shoemaker, and Aaron Zimmerman. Accuracy limitations of existing numerical relativity waveforms on the data analysis of current and future ground-based detectors. *Phys. Rev. D*, 110(2):024023, 2024. doi: 10.1103/PhysRevD.110.024023.
- [22] <https://www.black-holes.org/SpEC.html>.
- [23] Nils Deppe, William Throwe, Lawrence E. Kidder, Nils L. Vu, Kyle C. Nelli, Cristóbal Armaza, Marceline S. Bonilla, François Hébert, Yoonsoo Kim, Prayush Kumar, Geoffrey Lovelace, Alexandra Macedo, Jordan Moxon, Eamonn O’Shea, Harald P. Pfeiffer, Mark A. Scheel, Saul A. Teukolsky, Nikolas A. Wittek, Isha Anantpurkar, Carter Anderson, Michael Boyle, Alexander Carpenter, Andrea Ceja, Himanshu Chaudhary, Nicholas Corso, Nora Fayyazuddin Ljungberg, Francois Foucart, Noora Ghadiri, Matthew Giesler, Jason S. Guo, Sarah Habib, Chenhang Huang, Dante A. B. Iozzo, Ken Z. Jones, Guillermo Lara, Isaac Legred, Dongjun Li, Sizheng Ma, Denyz Melchor, Iago Mendes, Marlo Morales, Elias R. Most, Peter James Nee, Alejandro Osorio, Michael A. Pajkos, Kyle Pannone, Vaishak Prasad, Teresita Ramirez, Noah Ring, Hannes R. Rüter, Jennifer Sanchez, Leo C. Stein,

- Daniel Tellez, Sierra Thomas, Vittoria Tommasini, Daniel Vieira, Tom Włodarczyk, David Wu, and Jooheon Yoo. SpECTRE, April 2025. URL <https://doi.org/10.5281/zenodo.15259042>.
- [24] Hengrui Zhu, Jacob Fields, Francesco Zappa, David Radice, James M. Stone, Alireza Rashti, William Cook, Sebastiano Bernuzzi, and Boris Daszuta. Performance-portable numerical relativity with AthenaK. *Astrophys. J. Suppl.*, 278(2):50, 2025. doi: 10.3847/1538-4365/adcf96.
- [25] Swapnil Shankar, Philipp Mösta, Steven R. Brandt, Roland Haas, Erik Schnetter, and Yannick de Graaf. GRaM-X: A new GPU-accelerated dynamical spacetime GRMHD code for exascale computing with the Einstein Toolkit. *Class. Quant. Grav.*, 40(20):205009, 2023. doi: 10.1088/1361-6382/acf2d9.
- [26] Jay V. Kalinani et al. AsterX: A new open-source GPU-accelerated GRMHD code for dynamical spacetimes. *Class. Quant. Grav.*, 42(2):025016, 2025. doi: 10.1088/1361-6382/ad9c11.
- [27] Weiqun Zhang, Andrew Myers, Kevin Gott, Ann Almgren, and John Bell. AMReX: Block-structured adaptive mesh refinement for multiphysics applications. *The International Journal of High Performance Computing Applications*, 35(6):508–526, 2021. doi: 10.1177/10943420211022811. URL <https://doi.org/10.1177/10943420211022811>.
- [28] Christian R. Trott et al. Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, 33(4): 805–817, 2022. doi: 10.1109/TPDS.2021.3097283.
- [29] Wolfgang Tichy, Liwei Ji, Ananya Adhikari, Alireza Rashti, and Michal Pirog. The new discontinuous Galerkin methods based numerical relativity program Nmesh. *Class. Quant. Grav.*, 40(2):025004, 2023. doi: 10.1088/1361-6382/acaae7.
- [30] William Throwe and Saul Teukolsky. A high-order, conservative integrator with local time-stepping. *SIAM Journal on Scientific Computing*, 42(6): A3730–A3760, 2020. doi: 10.1137/19M1292692. URL <https://doi.org/10.1137/19M1292692>.
- [31] J. Davison de St. Germain, John McCorquodale, Steven G. Parker, and Christopher R. Johnson. Uintah: A massively parallel problem solving environment. In *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing*, HPDC '00, page 33, USA, 2000. IEEE Computer Society. ISBN 0769507832.
- [32] Kyle Burke. Chapel: A versatile language for teaching parallel programming: Conference workshop. *J. Comput. Sci. Coll.*, 30(6):16, June 2015. ISSN 1937-4771.

- [33] Laxmikant Kale et al. Uiuc-ppl/Charm: Charm++ version 6.10.2. 10.5281/zenodo.3972617, August 2020. URL <https://doi.org/10.5281/zenodo.3972617>.
- [34] Philippe Pébay, Janine C. Bennett, David Hollman, Sean Treichler, Patrick S. McCormick, Christine M. Sweeney, Hemanth Kolla, and Alex Aiken. Towards asynchronous many-task in situ data analysis using Legion. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1033–1037, 2016. doi: 10.1109/IPDPSW.2016.24.
- [35] Pierre Chevalier, Bartłomiej Kaminski, Fraser Hutchison, Qi Ma, Spandan Sharma, Andreas Fackler, and William J Buchanan. Protocol for asynchronous, reliable, secure and efficient consensus (PARSEC) version 2.0, July 2019. URL <https://arxiv.org/abs/1907.11445>.
- [36] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 5.0*, June 2025. URL <https://www.mpi-forum.org/docs/mpi-5.0/mpi50-report.pdf>.
- [37] Carsten Gundlach. Pseudospectral apparent horizon finders: An efficient new algorithm. *Phys. Rev. D*, 57:863–875, 1998. doi: 10.1103/PhysRevD.57.863.
- [38] Mark A. Scheel, Harald P. Pfeiffer, Lee Lindblom, Lawrence E. Kidder, Oliver Rinne, and Saul A. Teukolsky. Solving Einstein’s equations with dual coordinate frames. *Phys. Rev.*, D74:104006, 2006. doi: 10.1103/PhysRevD.74.104006.
- [39] Daniel A. Hemberger, Mark A. Scheel, Lawrence E. Kidder, Béla Szilágyi, Geoffrey Lovelace, Nicholas W. Taylor, and Saul A. Teukolsky. Dynamical excision boundaries in spectral evolutions of binary black hole spacetimes. *Class. Quant. Grav.*, 30:115001, 2013. doi: 10.1088/0264-9381/30/11/115001.
- [40] Lee Lindblom, Mark A. Scheel, Lawrence E. Kidder, Robert Owen, and Oliver Rinne. A new generalized harmonic evolution system. *Class. Quant. Grav.*, 23:S447–S462, 2006. doi: 10.1088/0264-9381/23/16/S09.
- [41] Bela Szilagyi, Lee Lindblom, and Mark A. Scheel. Simulations of binary black hole mergers using spectral methods. *Phys. Rev. D*, 80:124010, 2009. doi: 10.1103/PhysRevD.80.124010.
- [42] Geoffrey Lovelace, Kyle C. Nelli, et al. Simulating binary black hole mergers using discontinuous Galerkin methods. *Class. Quant. Grav.*, 42(3):035001, 2025. doi: 10.1088/1361-6382/ad9f19.
- [43] Luc Blanchet. Post-Newtonian theory for gravitational waves. *Living Rev. Rel.*, 27(1):4, 2024. doi: 10.1007/s41114-024-00050-z.

- [44] Yan Solihin. *Fundamentals of parallel multicore architecture*. Chapman & Hall/CRC Computational Science. Chapman & Hall/CRC, Philadelphia, PA, June 2020.
- [45] Michael J. Quinn. *Parallel computing (2nd ed.): Theory and practice*. McGraw-Hill, Inc., USA, 1994. ISBN 0070512949.
- [46] Nils Deppe et al. Binary neutron star mergers using a discontinuous Galerkin-finite difference hybrid method. *Class. Quant. Grav.*, 41(24):245002, 2024. doi: 10.1088/1361-6382/ad88cf.
- [47] Rezzolla, L. and Zanotti, O. *Relativistic Hydrodynamics*. Oxford University Press, September 2013. URL <http://adsabs.harvard.edu/abs/2013rehy.book.....R>.
- [48] Soham Mandal and Paul C. Duffell. SPROUT: A moving-mesh hydro code using a uniformly expanding cartesian grid. *The Astrophysical Journal Supplement Series*, 269(1):30, November 2023. doi: 10.3847/1538-4365/acfc19. URL <https://dx.doi.org/10.3847/1538-4365/acfc19>.
- [49] Sarah Habib et al. Error quantification and comparison of binary neutron star gravitational waveforms from numerical relativity codes. September 2025. URL <https://arxiv.org/abs/2509.23028>.
- [50] Guillermo Lara, Harald P. Pfeiffer, Nikolas A. Wittek, Nils L. Vu, Kyle C. Nelli, Alexander Carpenter, Geoffrey Lovelace, Mark A. Scheel, and William Throwe. Scalarization of isolated black holes in scalar Gauss-Bonnet theory in the fixing-the-equations approach. *Phys. Rev. D*, 110(2):024033, 2024. doi: 10.1103/PhysRevD.110.024033.
- [51] Nikolas A. Wittek, Leor Barack, Harald P. Pfeiffer, Adam Pound, Nils Deppe, Lawrence E. Kidder, Alexandra Macedo, Kyle C. Nelli, William Throwe, and Nils L. Vu. Relieving scale disparity in binary black hole simulations. October 2024. URL <https://arxiv.org/abs/2410.22290>.
- [52] Guillermo Lara et al. Signatures from metastable oppositely-charged black hole binaries in scalar Gauss-Bonnet gravity. May 2025. URL <https://arxiv.org/abs/2505.14785>.
- [53] Till Tantau. The TikZ and PGF packages. github.com/pgf-tikz/pgf. URL <https://github.com/pgf-tikz/pgf>.
- [54] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., Clifton Park, NY, USA, 2015. ISBN 1930934300.
- [55] James Ahrens, Berk Geveci, and Charles Law. ParaView: An end-user tool for large-data visualization. In *Visualization Handbook*. Butterworth-Heinemann, Burlington, 2005. ISBN 978-0123875822. doi: 10.1016/B978-012387582-2/

-
- 50038-1. URL <https://doi.org/10.1016/B978-012387582-2/50038-1>.

Chapter 3

SIMULATING BINARY BLACK HOLE MERGERS USING DISCONTINUOUS GALERKIN METHODS

Geoffrey Lovelace, Kyle C. Nelli, et al. Simulating binary black hole mergers using discontinuous Galerkin methods. *Class. Quant. Grav.*, 42(3):035001, 2025. doi: 10.1088/1361-6382/ad9f19.

Binary black holes are the most abundant source of gravitational-wave observations. Gravitational-wave observatories in the next decade will require tremendous increases in the accuracy of numerical waveforms modeling binary black holes, compared to today’s state of the art. One approach to achieving the required accuracy is using spectral-type methods that scale to many processors. Using the SpECTRE numerical-relativity code, we present the first simulations of a binary black hole inspiral, merger, and ringdown using discontinuous Galerkin methods. The efficiency of discontinuous Galerkin methods allows us to evolve the binary through ~ 18 orbits at reasonable computational cost. We then use SpECTRE’s Cauchy-Characteristic Evolution (CCE) code to extract the gravitational waves at future null infinity. The open-source nature of SpECTRE means this is the first time a spectral-type method for simulating binary black hole evolutions is available to the entire numerical-relativity community.

3.1 Introduction

Binary black holes are the most abundant source of gravitational-wave observations to date [1]. Realizing the scientific potential of these observations requires accurate models of the emitted gravitational waves as the black holes inspiral, merge, and ring down to a final, stationary state. Building these models requires numerical-relativity (NR) simulations of binary black holes, because analytic approximations (e.g. the post-Newtonian [2] approximation) alone break down near the time of merger.

Since the first breakthrough simulations [3–5], the NR community has developed codes capable of evolving two black holes through inspiral, merger, and ringdown (see [6, 7] for a review). Several groups have used NR codes to build catalogs of gravitational waveforms for applications to gravitational-wave astronomy [8–15]. Today’s NR codes are sufficiently accurate for the observations that LIGO and Virgo are making. However, observatories planned for the next decade, including the Einstein Telescope [16] and Cosmic Explorer [17] on Earth and the Laser Interferometer Space Antenna (LISA) [18] in space, will be so sensitive that they will require NR waveforms with a substantial increase in accuracy [19–21].

Spectral-type methods are extremely efficient; this makes them a promising avenue toward the ultimate goal of achieving the needed accuracy for future gravitational-wave observatories. In comparison, almost all current NR codes for evolving binary black holes use finite-difference methods, with numerical errors decreasing as a power law with increasing resolution. However, recent results from the AthenaK code [22] show that finite-difference methods using graphics processing units (GPUs) might be another approach to achieving the required accuracy. The Spectral Einstein Code (SpEC) [23] uses a pseudospectral method (see [24] for a review of these methods) to construct and evolve binary-black-hole initial data. With pseudospectral methods, errors decrease exponentially with increasing number of grid points in the computational domain’s elements (“ p -refinement”). SpEC’s exponential convergence makes it highly efficient, but its performance, and therefore the achievable accuracy, is limited by aspects of its design. For instance, because it uses computational domains divided into a small number of high-resolution elements, SpEC simulations of binary black holes cannot scale beyond $\mathcal{O}(10^2)$ CPU cores. SpEC is also a closed-source code, unavailable to most of the NR community. Other examples of pseudospectral or spectral methods being used for solving the initial value problem are Elliptica [25], FUKA [26], and bamps [27]. In terms of evolving spacetimes, the Nmesh [28] code has been used to success-

fully simulate single black holes using discontinuous Galerkin methods, and the `bamps` [29, 30] code uses pseudospectral methods to evolve spacetimes with single dynamical black holes with a focus on critical behavior [31–38] but has also simulated boson stars [39]. Recently, [40] performed a 0.5 orbit grazing collision of two black holes (a similar setup to [41]) using a finite volume grid in the strong field region and a discontinuous-Galerkin method in the wave zone.

We present the first simulations of a binary black hole inspiral, merger, and ringdown using a discontinuous Galerkin (DG) method [42] (see [43] for a review of DG). The efficiency of DG methods allows us to evolve the binary through ~ 18 orbits at reasonable computational cost: DG, being a spectral-type method, has exponential convergence with p -refinement. For context, 18 orbits is slightly less than the median (20) for binary-black-hole simulations in the SXS catalog [12] (which also uses a spectral-type method) but larger than almost all of the simulations in the RIT and Maya catalogs [14, 15] (which use finite-difference methods). We chose the length of `SpECTRE`’s first binary-black-hole simulation largely out of convenience, balancing a desire to demonstrate `SpECTRE`’s capability with minimizing turnaround time as we tested and fine-tuned our methods. We expect that using `SpECTRE` to simulate more orbits would be straightforward, without requiring changes to the code, although extending the length beyond 100 orbits would likely require implementing in `SpECTRE` similar techniques as those discussed in [44], which presents a 175-orbit `SpEC` binary-black-hole simulation.

Specifically, in this work, we present `SpECTRE`’s [45] first simulations of ~ 18 orbits of inspiral, merger, and ringdown of an equal-mass, non-spinning binary black hole, using DG methods. We then use `SpECTRE`’s Cauchy Characteristic Evolution module [45–47] to evolve the gravitational waves to future null infinity. These results demonstrate that DG methods can provide high-accuracy gravitational waveforms from binary black hole mergers for application to gravitational-wave data analysis. By implementing our approach in `SpECTRE`, an open-source NR code, we are also making a spectral-type binary-black-hole evolution code available to the entire NR community for the first time.

The rest of this paper is organized as follows. In § 3.2, we discuss the numerical methods used in `SpECTRE`’s binary-black-hole simulations. Then, in § 3.3, we first test our method’s stability with simulations of single black holes before presenting results for simulations of binary black holes with `SpECTRE`. We briefly conclude in § 3.4.

3.2 Methods

3.2.1 Equations of Motion

We adopt the standard 3+1 form of the spacetime metric, (see, e.g., [48, 49]),

$$ds^2 = g_{ab}dx^a dx^b = -\alpha^2 dt^2 + \gamma_{ij} (dx^i + \beta^i dt) (dx^j + \beta^j dt), \quad (3.1)$$

where α is the lapse, β^i the shift vector, and γ_{ij} is the spatial metric. We use the Einstein summation convention, summing over repeated indices. Latin indices from the first part of the alphabet $a, b, c, \dots$ denote spacetime indices ranging from 0 to 3, while Latin indices $i, j, \dots$ are purely spatial, ranging from 1 to 3. We work in units where $c = G = 1$.

We evolve the first-order generalized harmonic (FOGH) system, given by [50],

$$\partial_t g_{ab} = (1 + \gamma_1) \beta^k \partial_k g_{ab} - \alpha \Pi_{ab} - \gamma_1 \beta^i \Phi_{iab} + \gamma_1 v_g^k (\partial_k g_{ab} - \Phi_{kab}), \quad (3.2)$$

$$\begin{aligned} \partial_t \Phi_{iab} = & \beta^k \partial_k \Phi_{iab} - \alpha \partial_i \Pi_{ab} + \alpha \gamma_2 \partial_i g_{ab} + \frac{1}{2} \alpha n^c n^d \Phi_{icd} \Pi_{ab} \\ & + \alpha \gamma^{jk} n^c \Phi_{ijc} \Phi_{kab} - \alpha \gamma_2 \Phi_{iab}, \end{aligned} \quad (3.3)$$

$$\begin{aligned} \partial_t \Pi_{ab} = & \beta^k \partial_k \Pi_{ab} - \alpha \gamma^{ki} \partial_k \Phi_{iab} + \gamma_1 \gamma_2 \beta^k \partial_k g_{ab} \\ & + 2\alpha g^{cd} (\gamma^{ij} \Phi_{ica} \Phi_{jdb} - \Pi_{ca} \Pi_{db} - g^{ef} \Gamma_{ace} \Gamma_{bdf}) \\ & - 2\alpha \nabla_{(a} H_{b)} - \frac{1}{2} \alpha n^c n^d \Pi_{cd} \Pi_{ab} - \alpha n^c \Pi_{ci} \gamma^{ij} \Phi_{jab} \\ & + \alpha \gamma_0 (2\delta^c_{(a} n_{b)} - g_{ab} n^c) \mathcal{C}_c - \gamma_1 \gamma_2 \beta^i \Phi_{iab}, \end{aligned} \quad (3.4)$$

where g_{ab} is the spacetime metric, $\Phi_{iab} = \partial_i g_{ab}$, $\Pi_{ab} = n^c \partial_c g_{ab}$, n^a is the unit normal vector to the spatial slice, γ_0 damps the 1-index or gauge constraint $\mathcal{C}_a = H_a + \Gamma_a$, γ_1 controls the linear degeneracy of the system, γ_2 damps the 3-index constraint $\mathcal{C}_{iab} = \partial_i g_{ab} - \Phi_{iab}$, Γ_{abc} are the spacetime Christoffel symbols of the first kind, $\Gamma_a = g^{bc} \Gamma_{bca}$, and v_g^k is the grid/mesh velocity as discussed in § 3.2.2.

The gauge source function H_a can be any arbitrary function depending only upon the spacetime coordinates x^a and g_{ab} , but not derivatives of g_{ab} , since that may spoil the strong hyperbolicity of the system [51, 52].

Defining s_i to be the unit normal covector to a 2d surface with $s_a = (0, s_i)$, and $s^a = g^{ab} s_b$, the characteristic fields for the FOGH system are [50]

$$w_{ab}^g = g_{ab}, \quad (3.5)$$

$$w_{iab}^0 = (\delta_i^k - s^k s_i) \Phi_{kab}, \quad (3.6)$$

$$w_{ab}^\pm = \Pi_{ab} \pm s^i \Phi_{iab} - \gamma_2 g_{ab}, \quad (3.7)$$

with associated characteristic speeds

$$\lambda_{w^g} = - (1 + \gamma_1) \beta^i s_i - (1 + \gamma_1) v_g^i s_i, \quad (3.8)$$

$$\lambda_{w^0} = - \beta^i s_i - v_g^i s_i, \quad (3.9)$$

$$\lambda_{w^\pm} = \pm \alpha - \beta^i s_i - v_g^i s_i, \quad (3.10)$$

where we denote the velocity of the grid/mesh as v_g^i (see § 3.2.2 for details on our moving mesh method). The evolved variables as a function of the characteristic fields are given by

$$g_{ab} = w_{ab}^g, \quad (3.11)$$

$$\Pi_{ab} = \frac{1}{2} (w_{ab}^+ + w_{ab}^-) + \gamma_2 w_{ab}^g, \quad (3.12)$$

$$\Phi_{iab} = \frac{1}{2} (w_{ab}^+ - w_{ab}^-) s_i + w_{iab}^0. \quad (3.13)$$

The constraints for the FOGH system are [50]

$$\mathcal{C}_a = H_a + \Gamma_a, \quad (3.14)$$

$$\begin{aligned} \mathcal{C}_{ia} = & \gamma^{jk} \partial_j \Phi_{ika} - \frac{1}{2} \gamma_a^j g^{cd} \partial_j \Phi_{icd} + n^b \partial_i \Pi_{ba} - \frac{1}{2} n_a g^{cd} \partial_i \Pi_{cd} \\ & + \partial_i H_a + \frac{1}{2} \gamma_a^j \Phi_{jcd} \Phi_{ief} g^{ce} g^{df} + \frac{1}{2} \gamma^{jk} \Phi_{jcd} \Phi_{ike} g^{cd} n^e n_a \\ & - \gamma^{jk} \gamma^{mn} \Phi_{jma} \Phi_{ikn} + \frac{1}{2} \Phi_{icd} \Pi_{be} n_a \left(g^{cb} g^{de} + \frac{1}{2} g^{be} n^c n^d \right) \\ & - \Phi_{icd} \Pi_{ba} n^c \left(g^{bd} + \frac{1}{2} n^b n^d \right) + \frac{1}{2} \gamma_2 (n_a g^{cd} - 2 \delta_a^c n^d) \mathcal{C}_{icd}, \end{aligned} \quad (3.15)$$

$$\mathcal{C}_{iab} = \partial_i g_{ab} - \Phi_{iab}, \quad (3.16)$$

$$\mathcal{C}_{ijab} = \partial_i \Phi_{jab} - \partial_j \Phi_{iab}, \quad (3.17)$$

and

$$\begin{aligned}
\mathcal{F}_a = & \frac{1}{2}\gamma_a^i g^{bc} \partial_i \Pi_{bc} - \gamma^{ij} \partial_i \Pi_{ja} - \gamma^{ij} n^b \partial_i \Phi_{jba} + \frac{1}{2} n_a g^{bc} \gamma^{ij} \partial_i \Phi_{jbc} \\
& + n_a \gamma^{ij} \partial_i H_j + \gamma_a^i \Phi_{ijb} \gamma^{jk} \Phi_{kcd} g^{bd} n^c - \frac{1}{2} \gamma_a^i \Phi_{ijb} \gamma^{jk} \Phi_{kcd} g^{cd} n^b \\
& - \gamma_a^i n^b \partial_i H_b + \gamma^{ij} \Phi_{icd} \Phi_{jba} g^{bc} n^d - \frac{1}{2} n_a \gamma^{ij} \gamma^{mn} \Phi_{imc} \Phi_{njd} g^{cd} \\
& - \frac{1}{4} n_a \gamma^{ij} \Phi_{icd} \Phi_{jbe} g^{cb} g^{de} + \frac{1}{4} n_a \Pi_{cd} \Pi_{be} g^{cb} g^{de} - \gamma^{ij} H_i \Pi_{ja} \\
& - n^b \gamma^{ij} \Pi_{bi} \Pi_{ja} - \frac{1}{4} \gamma_a^i \Phi_{icd} n^c n^d \Pi_{be} g^{be} + \frac{1}{2} n_a \Pi_{cd} \Pi_{be} g^{ce} n^d n^b \\
& + \gamma_a^i \Phi_{icd} \Pi_{be} n^c n^b g^{de} - \gamma^{ij} \Phi_{iba} n^b \Pi_{je} n^e - \frac{1}{2} \gamma^{ij} \Phi_{icd} n^c n^d \Pi_{ja} \\
& - \gamma^{ij} H_i \Phi_{jba} n^b + \gamma_a^i \Phi_{icd} H_b g^{bc} n^d + \gamma_2 \left(\gamma^{id} \mathcal{C}_{ida} - \frac{1}{2} \gamma_a^i g^{cd} \mathcal{C}_{icd} \right) \\
& + \frac{1}{2} n_a \Pi_{cd} g^{cd} H_b n^b - n_a \gamma^{ij} \Phi_{ijc} H_d g^{cd} + \frac{1}{2} n_a \gamma^{ij} H_i \Phi_{jcd} g^{cd}. \tag{3.18}
\end{aligned}$$

While only the gauge constraint (3.14) and 3-index constraint (3.16) are damped, all constraints can be monitored to check the accuracy of the numerical simulation. All the constraints can be combined into a scalar, the constraint energy, given by [50]

$$\begin{aligned}
\mathcal{E} = & \delta^{ab} [\mathcal{C}_a \mathcal{C}_b + (\mathcal{F}_a \mathcal{F}_b + \mathcal{C}_{ia} \mathcal{C}_{jb} \gamma^{ij})] \\
& + \delta^{ab} \delta^{cd} (\mathcal{C}_{iac} \mathcal{C}_{jbd} \gamma^{ij} + \mathcal{C}_{ikac} \mathcal{C}_{jlbd} \gamma^{ij} \gamma^{kl}). \tag{3.19}
\end{aligned}$$

In practice we have also found that it is typically only necessary to monitor violations of the constraints $\mathcal{C}_a$ and $\mathcal{C}_{iab}$, because they typically grow first and the other violations grow as a consequence.

3.2.2 Discontinuous Galerkin (DG) Method

SpECTRE uses a DG method for the spatial discretization. We refer readers to [53, 54] and references therein for a detailed discussion of the method and its implementation in SpECTRE; here we summarize the necessary results. The FOGH equations are a first-order strongly hyperbolic system in non-conservative form, which takes the general form

$$\partial_t u_\alpha + B_{\alpha\beta}^i(u_\alpha) \partial_i u_\beta = S_\alpha(u_\alpha), \tag{3.20}$$

where $u_\alpha = \{g_{ab}, \Phi_{iab}, \Pi_{ab}\}$ is the state vector of evolved variables, $B_{\alpha\beta}^i(u)$ is a matrix that depends only on u_α , and $S_\alpha(u)$ are source terms that also only depend on u_α . We denote the logical coordinates of our Legendre-Gauss-Lobatto DG scheme

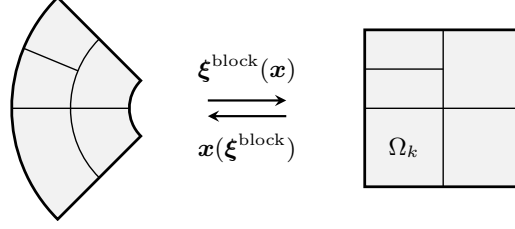


Figure 3.1: A diagram of the forward and reverse mappings $x^i(\hat{t}, \xi^i)$ and $\xi^i(t, x^i)$, respectively, from logical (right side of the diagram) to inertial coordinates (left side of the diagram) for an element Ω_k .

by $\{\hat{t}, \xi^i\} = \{\hat{t}, \xi, \eta, \zeta\}$ and the inertial coordinates as $\{t = \hat{t}, x^i(\hat{t}, \xi^i)\}$. We are using a moving mesh as in [53, 55]; therefore, the mapping from logical to inertial coordinates is time dependent. We denote the determinant of the spatial Jacobian of this map as

$$J = \det \left(\frac{\partial x^i}{\partial \xi^i} \right), \quad (3.21)$$

and the grid or mesh velocity by [53, 55]

$$v_g^i = \partial_{\hat{t}} x^i. \quad (3.22)$$

SpECTRE decomposes the domain into a set of non-overlapping hexahedra which are deformed using the map $x^i(\hat{t}, \xi^i)$ illustrated in Fig. 3.1. SpECTRE uses a different number of grid points in each logical direction, which we denote by $N_{\hat{\xi}}$ in the ξ direction, $N_{\hat{\eta}}$ in the η direction, and $N_{\hat{\zeta}}$ in the ζ direction below.

Since we use a moving mesh, we evolve the system

$$\partial_{\hat{t}} u_{\alpha} + [B_{\alpha\beta}^i(u_{\alpha}) - v_g^i \delta_{\alpha\beta}] \partial_i u_{\beta} = S(u_{\alpha}). \quad (3.23)$$

We denote grid point and modal indices with a breve, i.e. $u_{\hat{i}\hat{j}\hat{k}}$ is the value of u at

the grid point $(\check{i}, \check{j}, \check{k})$. The semi-discrete equations are given by [53, 55]

$$\begin{aligned}
 (\partial_t u_\alpha)_{\check{i}\check{j}\check{k}} = & (S_\alpha)_{\check{i}\check{j}\check{k}} - (B_{\alpha\beta}^i - v_g^i \delta_{\alpha\beta})_{\check{i}\check{j}\check{k}} \left[\left(\frac{\partial \xi^{\hat{1}}}{\partial x^i} \right)_{\check{i}\check{j}\check{k}} \sum_{\check{l}} D_{\check{i}\check{l}}^{(\hat{1})}(u_\beta)_{\check{l}\check{j}\check{k}} \right. \\
 & - \left(\frac{\partial \xi^{\hat{2}}}{\partial x^i} \right)_{\check{i}\check{j}\check{k}} \sum_{\check{l}} D_{\check{j}\check{l}}^{(\hat{2})}(u_\beta)_{\check{i}\check{l}\check{k}} + \left. \left(\frac{\partial \xi^{\hat{3}}}{\partial x^i} \right)_{\check{i}\check{j}\check{k}} \sum_{\check{l}} D_{\check{k}\check{l}}^{(\hat{3})}(u_\beta)_{\check{i}\check{j}\check{l}} \right] \\
 & - \frac{\delta_{N_{\check{k}}\check{k}}}{w_{\check{k}} J_{\check{i}\check{j}\check{k}}} \left\{ \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{\check{i}\check{j}N_{\check{k}}} + \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{\check{i}\check{j}0} \right\} \\
 & - \frac{\delta_{N_{\check{j}}\check{j}}}{w_{\check{j}} J_{\check{i}\check{j}\check{k}}} \left\{ \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{iN_{\check{j}}\check{k}} + \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{i0\check{k}} \right\} \\
 & - \frac{\delta_{N_{\check{i}}\check{i}}}{w_{\check{i}} J_{\check{i}\check{j}\check{k}}} \left\{ \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{N_{\check{i}}\check{j}\check{k}} + \left[J \sqrt{\frac{\partial \xi^{\hat{3}}}{\partial x^i} \gamma^{ij} \frac{\partial \xi^{\hat{3}}}{\partial x^j}} D_\alpha \right]_{0\check{j}\check{k}} \right\}, \tag{3.24}
 \end{aligned}$$

where w_i are the Legendre-Gauss-Lobatto integration weights. We use a method of lines approach to integrate these in time, with the details discussed in § 3.2.5 below.

For the boundary terms D_α , we use an upwind multi-penalty method [24, 56–58] given by

$$D_{gab} = \tilde{\lambda}_{wg}^{\text{ext}} w_{ab}^{\text{ext},g} - \tilde{\lambda}_{wg}^{\text{int}} w_{ab}^{\text{int},g}, \tag{3.25}$$

$$\begin{aligned}
 D_{\Pi_{ab}} = & \frac{1}{2} \left(\tilde{\lambda}_{w^+}^{\text{ext}} w_{ab}^{\text{ext},+} + \tilde{\lambda}_{w^-}^{\text{ext}} w_{ab}^{\text{ext},-} \right) + \tilde{\lambda}_{wg}^{\text{ext}} \gamma_2 w_{ab}^{\text{ext},g} \\
 & - \frac{1}{2} \left(\tilde{\lambda}_{w^+}^{\text{int}} w_{ab}^{\text{int},+} + \tilde{\lambda}_{w^-}^{\text{int}} w_{ab}^{\text{int},-} \right) - \tilde{\lambda}_{wg}^{\text{int}} \gamma_2 w_{ab}^{\text{int},g}, \tag{3.26}
 \end{aligned}$$

$$\begin{aligned}
 D_{\Phi_{iab}} = & \frac{1}{2} \left(\tilde{\lambda}_{w^+}^{\text{ext}} w_{ab}^{\text{ext},+} - \tilde{\lambda}_{w^-}^{\text{ext}} w_{ab}^{\text{ext},-} \right) s_i^{\text{ext}} + \tilde{\lambda}_{w^0}^{\text{ext}} w_{iab}^{\text{ext},0} \\
 & - \frac{1}{2} \left(\tilde{\lambda}_{w^+}^{\text{int}} w_{ab}^{\text{int},+} - \tilde{\lambda}_{w^-}^{\text{int}} w_{ab}^{\text{int},-} \right) s_i^{\text{int}} - \tilde{\lambda}_{w^0}^{\text{int}} w_{iab}^{\text{int},0}, \tag{3.27}
 \end{aligned}$$

where the spatial normal vector to the element interface s_i^{int} is pointing out of the DG element and $\tilde{\lambda} = 0$ if $\lambda > 0$, otherwise $\tilde{\lambda} = \lambda$, i.e. $\tilde{\lambda} = \lambda \Theta(-\lambda)$. Note that we assume s_i^{ext} and s_i^{int} point in the same direction. Also note that these boundary flux terms differ from the multi-penalty approach used in SpEC by a factor of 2. That is,

$$D^{\text{SpECTRE}} = 2D^{\text{SpEC}}, \tag{3.28}$$

ultimately because the lifting terms are different. In SpEC and, similarly in bamps[29], the penalty term is derived from requiring that the total energy be non-increasing,

while in SpECTRE the terms come from an integration by parts when deriving the semi-discrete DG equations.

3.2.3 Boundary Conditions

At the outer radial boundary, we apply constraint-preserving boundary conditions [50, 59] by adding terms to the time derivative of the characteristic fields and thus also the time derivatives of the evolved variables. We use the characteristic fields and speeds defined in § 3.2.1. We define $d_t w_{\hat{\alpha}}$ as the time derivatives substituted into the transformation equations to the characteristic fields. That is,

$$d_t w_{ab}^g = \partial_t g_{ab}, \quad (3.29)$$

$$d_t w_{iab}^0 = (\delta_i^k - s^k s_i) \partial_t \Phi_{kab}, \quad (3.30)$$

$$d_t w_{ab}^{\pm} = \partial_t \Pi_{ab} \pm s^i \partial_t \Phi_{iab} - \gamma_2 \partial_t g_{ab}. \quad (3.31)$$

We also define $D_t w_{\hat{\alpha}}$ as the characteristic field transformation of the volume right-hand-side, i.e. $\partial_t u_{\alpha}$ *without* any boundary terms. Finally, for brevity we define the projection tensor $P_{ab} = g_{ab} + n_a n_b - s_a s_b$, the inward directed null vector field $k^a = (n^a - s^a)/\sqrt{2}$, and the outgoing null vector field $l^a = (n^a + s^a)/\sqrt{2}$.

The fields $d_t w_{ab}^g$ and $d_t w_{iab}^0$ are determined solely by the constraint-preserving boundary condition, while the boundary condition for $d_t w_{ab}^-$ is composed of three parts: the constraint preserving part, the physical part, and the gauge part. We denote these as B_{ab}^C , B_{ab}^P and B_{ab}^G . With this, the boundary conditions imposed on the fields are

$$d_t w_{ab}^g = D_t w_{ab}^g + \lambda_{w^g} s^i \mathcal{C}_{iab}, \quad (3.32)$$

$$d_t w_{kab}^0 = D_t w_{kab}^0 + \lambda_{w^0} s^i P_k^j \mathcal{C}_{ijab}, \quad (3.33)$$

$$d_t w_{ab}^- = D_t w_{ab}^- + \lambda_{w^-} [B_{ab}^C + B_{ab}^P + B_{ab}^G]. \quad (3.34)$$

Transforming to the evolved variables we find that the following terms need to be added in order to impose the boundary condition,

$$\partial_t g_{ab} \rightarrow \partial_t g_{ab} + \lambda_{w^g} s^i \mathcal{C}_{iab}, \quad (3.35)$$

$$\partial_t \Pi_{ab} \rightarrow \partial_t \Pi_{ab} + \frac{1}{2} \lambda_{w^-} [B_{ab}^C + B_{ab}^P + B_{ab}^G] + \gamma_2 \lambda_{w^g} s^i \mathcal{C}_{iab}, \quad (3.36)$$

$$\partial_t \Phi_{iab} \rightarrow \partial_t \Phi_{iab} - \frac{s_i}{2} \lambda_{w^-} [B_{ab}^C + B_{ab}^P + B_{ab}^G] + \lambda_{w^0} s^i P_k^j \mathcal{C}_{ijab}. \quad (3.37)$$

We now need to specify the B_{ab} boundary conditions. The constraint-preserving part is

$$B_{ab}^C = \sqrt{2} \left(\frac{1}{2} P_{ab} l^c + \frac{1}{2} l_a l_b k^c - l_{(a} P_{b)}^c \right) (\mathcal{F}_c - s^k \mathcal{C}_{kc}). \quad (3.38)$$

The physical boundary conditions are determined by the propagating parts of the Weyl curvature tensor. That is,

$$B_{ab}^P = \left(P_a^c P_b^d - \frac{1}{2} P_{ab} P^{cd} \right) [C_{cd}^- - \gamma_2 s^i \mathcal{C}_{icd}], \quad (3.39)$$

where C_{ab}^- is the inward propagating part of the Weyl tensor, given by

$$C_{ab}^\pm = \left(P_a^c P_b^d - \frac{1}{2} P_{ab} P^{cd} \right) (n^e \mp s^e) (n^f \mp s^f) C_{cdf}. \quad (3.40)$$

For the simulations presented here, we set $C_{ab}^- = 0$, though Cauchy-Characteristic matching [60] can be used to prescribe a more physically motivated boundary condition. Recently [61] presented an alternative approach to Cauchy-Characteristic matching for providing high-order non-reflecting boundary conditions. Finally, the gauge boundary condition is set using a Sommerfeld condition on the components not set by the constraint-preserving and physical boundary conditions. The projector for the gauge boundary condition is given by

$$\begin{aligned} \delta_a^c \delta_b^d - P_{ab}^{Ccd} - P_{ab}^{Pcd} &= \delta_a^c \delta_b^d - \frac{1}{2} P_{ab} P^{cd} + 2l_{(a} P_{b)}^{(c} k^{d)} \\ &\quad - l_a l_b k^c k^d - P_a^c P_b^d + \frac{1}{2} P_{ab} P^{cd} \\ &= \delta_a^c \delta_b^d + 2l_{(a} P_{b)}^{(c} k^{d)} - l_a l_b k^c k^d - P_a^c P_b^d. \end{aligned} \quad (3.41)$$

The Sommerfeld condition is

$$B_{ab}^G = \frac{1}{\lambda_{w-}} (2l_{(a} P_{b)}^{(c} k^{d)} - 2k_{(a} l_{b)} k^{(c} l^{d)} - k_a k_b l^c l^d) \left(\gamma_2 - \frac{1}{r} \right) \partial_t g_{cd}. \quad (3.42)$$

When evolving spacetimes with black holes, we excise the interior of the black hole as is done in SpEC [50]. At excision boundaries, all information is flowing out of the grid and into the black hole, so no boundary condition needs to be applied. However, we monitor the characteristic speeds, (3.8-3.10), and terminate the code if any of them point into the computational domain. We denote the radius of the excision surfaces by r_{exc} . See § 3.2.7 for a brief explanation of how we control r_{exc} to avoid any characteristic speed pointing into the computational domain.

3.2.4 Spectral Filter

We use an exponential filter applied to the spectral coefficient $c_{\check{i}}$ in order to eliminate aliasing-driven instabilities. Specifically, for a 1d spectral expansion

$$u(x) = \sum_{\check{i}=0}^N c_{\check{i}} P_{\check{i}}(x), \quad (3.43)$$

where $P_{\check{i}}(x)$ are the Legendre polynomials, we use the filter

$$c_{\check{i}} \rightarrow c_{\check{i}} \exp \left[-a \left(\frac{\check{i}}{N} \right)^{2b} \right]. \quad (3.44)$$

We choose the parameters $a = 64$ and $b = 210$ so that only the highest spectral mode is filtered. We apply the filter to all FOGH variables g_{ab} , Φ_{iab} and Π_{ab} . Note that the filter drops the order of convergence for the FOGH variables from $\mathcal{O}(N+1)$ to $\mathcal{O}(N)$ on the DG grid, but is necessary for stability.

3.2.5 Time Integration

We decompose the system using the method of lines and solve the resulting differential equations using a local adaptive time-stepper based on the Adams-Moulton predictor-corrector method [62]. The step size in each element is chosen based on an estimate of the truncation error of the time step, using the algorithm described in [63] §17.2.1. The specific values for the absolute and relative tolerances are given in § 3.3. As the time-stepping algorithm is more efficient for aligned steps of the same size, the step size in each element is rounded down to a value of the form $0.1M/2^n$ for some non-negative integer n . For the highest-resolution binary-black-hole run in § 3.3.3, this results in the most-demanding element taking 2^6 – 2^8 steps for each step on the least demanding element for most of the inspiral. At the time of merger, this can increase to as high as 2^{11} steps for the most-demanding element for each step on the least demanding element.

3.2.6 Gauge Condition

We evolve binary black holes (§ 3.3.3) using the Damped Harmonic gauge condition [52?]:

$$H_a = [\mu_{L1} \log(\sqrt{\gamma}/\alpha) + \mu_{L2} \log(1/\alpha)] n_a - \mu_S g_{ai} \beta^i / \alpha, \quad (3.45)$$

using

$$\mu_{L1} = A_{L1} e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_{L1}}, \quad (3.46)$$

$$\mu_{L2} = A_{L2} e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_{L2}}, \quad (3.47)$$

$$\mu_S = A_S e^{-(r/\sigma_r)^2} [\log(\sqrt{\gamma}/\alpha)]^{e_S}, \quad (3.48)$$

where r is the coordinate distance from the origin. This condition is designed to drive $\sqrt{\gamma}$ and α to one, while damping out oscillations in the shift. This is because we observe an explosive growth in $\sqrt{\gamma}$ and a rapid collapse in α as the black holes merge. In practice, this ensures coordinates remain sufficiently well behaved throughout inspiral, merger, and ringdown. The amplitudes A_{L1} , A_{L2} , and A_S and exponents e_{L1} , e_{L2} and e_S control the amount of damping, and the spatial decay width σ_r ensures that at large distances, the gauge reduces to harmonic gauge (i.e., to $H_a = 0$). In this paper, we choose $A_{L1} = A_S = 1$, $A_{L2} = 0$, $e_{L1} = e_{L2} = e_S = 2$, and $\sigma_r = 100/\sqrt{34.54}$. This choice for σ_r ensures that the spatial decay Gaussian falls to 10^{-15} at a distance $r = 100$ from the origin.

For some of the single black-hole evolutions (§ 3.3.1), we instead choose H_a to be Γ_a of the analytic initial data. For other single black-hole evolutions, we evolve in harmonic gauge, setting $H_a = 0$ everywhere. For the gauge wave evolution (§ 3.3.2), we set $H_a(t, x^i)$ to the value of $\Gamma_a(t, x^i)$ of the gauge wave analytic solution.

3.2.7 Control Systems

When evolving the FOGH system, if there are black holes, the physical singularities inside of the black holes must be excised from the computational domain. To position the excisions with our moving mesh (described in § 3.2.2), we use a feedback control system similar to what is presented in [55] and [64]. As discussed in § 3.2.3, the excision surfaces must have all characteristic speeds pointing out of the computational domain, so that no boundary condition must be imposed. In practice this means that the excision surfaces must remain inside the apparent horizons, with the caveat that having them too close to the singularity causes instabilities. In practice the excision surfaces are kept at approximately 95-99% of the apparent horizons' radii.

Since we *a priori* do not know the motion or shape of the apparent horizons, we use control theory to dynamically update the parameters of the moving mesh periodically during the simulation. The time-dependent coordinate maps of the moving mesh and control signals used to update them are discussed in [64] in §4.1-4.3, §4.5, and §5 for the inspiral and §6 for the ringdown.

The details of how the control systems are implemented within the context of asynchronous task-based parallelism along with the local adaptive time stepping described in § 3.2.5 are described in [65].

3.3 Results

In this section, we begin by testing SpECTRE’s long-term stability and convergence; first with evolutions of single black holes in different coordinate systems (§ 3.3.1) and then with an evolution of a time-dependent gauge wave on a flat spacetime background (§ 3.3.2). Finally, we present results from a complete simulation of the inspiral, merger, and ringdown of two black holes (§ 3.3.3). The SpECTRE input files used for simulations, including generating the BBH initial data, are provided as ancillary material with the paper.

3.3.1 Single Black Hole Evolutions

In this section, we use SpECTRE to evolve a single, stationary, black hole that, unless otherwise noted, is non-spinning. We evolve a black hole from analytic initial data corresponding to a black hole at rest centered at the origin with zero spin. We choose the mass of the black hole to be $M = 1$ and work in units of M . In each evolution we use the following values for the FOGH constraint damping parameters,

$$\gamma_0 = \gamma_2 = A_0 e^{-r^2/w_0^2} + A_1 e^{-r^2/w_1^2}, \quad (3.49)$$

$$\gamma_1 = -1, \quad (3.50)$$

where r is the coordinate distance from the origin, $A_0 = 7.0/M$, $A_1 = 0.1/M$, $w_0 = 2.5M$, and $w_1 = 100.0M$. The computational domain of each evolution covers a spherical shell volume (Fig. 3.2) with inner radius $r_{\text{in}} = r_{\text{exc}}$ which differs for our different test cases, and outer boundary coordinate radius $r_{\text{out}} = 1,000M$. We apply boundary conditions as described in § 3.2.3. We use a fourth-order Adams-Moulton predictor-corrector time integrator with absolute and relative time stepper tolerances of 10^{-8} and 10^{-6} , respectively, unless otherwise stated.

3.3.1.1 Kerr-Schild Coordinates

We first evolve a single black hole in Kerr-Schild coordinates from Kerr-Schild initial data. The inner radius of the computational domain is $r_{\text{in}} = r_{\text{exc}} = 1.9M$. In this case there are no coordinate dynamics, so a feedback control system is not necessary, though it is enabled in the simulations presented here. The left panel of Fig. 3.3 shows the gauge constraint $\mathcal{C}_a$ and the 3-index constraint $\mathcal{C}_{iab}$ as a function

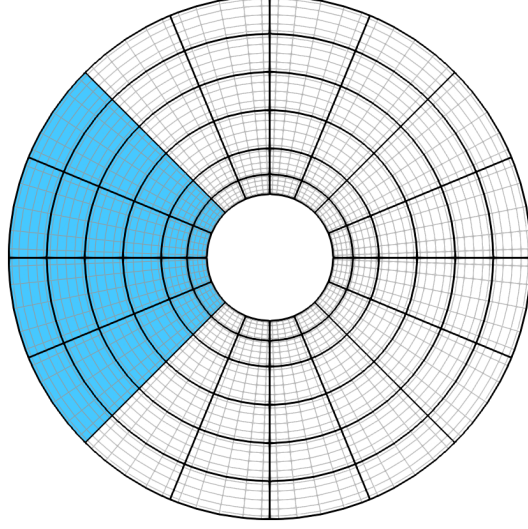


Figure 3.2: An illustration of a slice through the computational domain used in the evolutions of single black holes described in this paper. Six regions, each in the shape of a deformed cube, combine to cover the volume of a spherical shell, with the inner boundary (the excision surface) inside the black hole’s apparent horizon. The six regions are themselves refined radially and angularly into smaller deformed cubes. One of the six regions is shown in blue for clarity.

of time for several different resolutions. We evolve the lowest resolution to time $t = 10,000M$ to assess long-term stability, and we evolve the medium and high resolutions to assess convergence. To limit the computational cost of these tests, we choose to evolve the medium and high resolutions only to time $t = 2,000M$. All simulations are stable to time $t = 2000M$, and the lowest resolution remains stable to $t = 10,000M$. The amount of violation of the gauge constraint $\mathcal{C}_a$ and the 3-index constraint $\mathcal{C}_{iab}$ is indicative of the overall constraint violation in the simulation. In this evolution, the constraints remain approximately constant, and they decrease exponentially with increasing p -refinement (that is, increasing points per cell per dimension), as expected. We suspect the transient at $t = 2000M$ results from constraint violations reflecting off the outer boundary back to the interior.

The right panel of Fig. 3.3 demonstrates long-term stability and convergence for the same setup but with a black hole of dimensionless spin $\chi \equiv S/M^2 = 0.8$ (with $r_{\text{in}} = r_{\text{exc}} = 1.57M$). Again, we see that the constraints remain approximately constant and converge exponentially with increasing p -refinement.

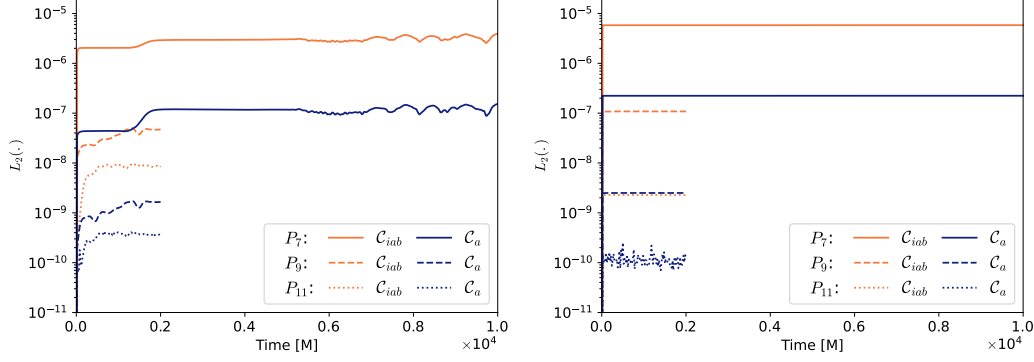


Figure 3.3: Constraint violations for single black hole evolutions in Kerr-Schild coordinates with Kerr-Schild initial data. We evolve the lowest resolution to time $t = 10,000M$ to demonstrate long-term stability and evolve two higher resolutions to time $t = 2,000M$ to assess convergence with spatial resolution. *Left*: Non-spinning black hole. *Right*: Spinning black-hole with dimensionless spin $\chi = S/M^2 = 0.8$.

3.3.1.2 Harmonic Coordinates

Next, we evolve a single black hole in harmonic gauge $H_a = 0$ using initial data also in harmonic gauge. Here, $r_{\text{in}} = r_{\text{exc}} = 0.9M$. The absolute and relative time stepper tolerances for the highest resolution of this test case are 10^{-10} and 10^{-8} , respectively. Again, since the initial data and evolution use the same gauge there are no gauge dynamics. The left panel of Fig. 3.4 shows the gauge constraint $\mathcal{C}_a$ and the 3-index constraint $\mathcal{C}_{iab}$ as a function of time for several different resolutions. We evolve the lowest resolution to time $t = 10,000M$ to assess stability and two higher resolutions to time $t = 5,000M$ to assess convergence. The constraints again remain constant, and they decrease exponentially with increasing p -refinement.

As a first test of the control system, we evolve a Kerr-Schild black hole in harmonic coordinates. The inner radius of the domain again is $r_{\text{in}} = r_{\text{exc}} = 1.8M$. The differing gauge choices in the initial data and evolution create coordinate dynamics that cause the BH horizon to shrink. The control system (§ 3.2.7) must decrease the radius of the excision surface smoothly and precisely to avoid incoming characteristic speeds, so that the problem remains well-posed and the code does not terminate. The right panel of Fig. 3.4 shows constraint violations over time for three resolutions. All evolutions are stable, and the constraint violations converge away. However, the constraints remain larger until after time $1,000M$. We suspect this is caused by initial gauge dynamics, i.e., by time-dependent, outward-moving coordinate effects that travel outward until exiting the domain through the outer boundary at

$r_{\text{out}} = 1,000M$.

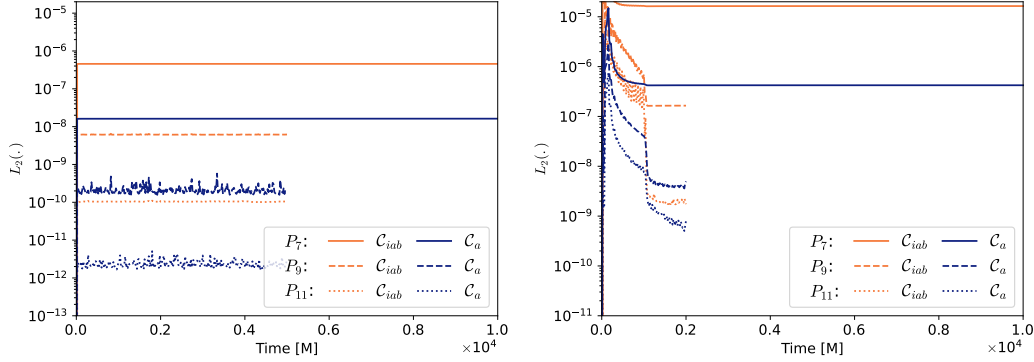


Figure 3.4: Constraint violations for single black hole evolutions in harmonic coordinates. We evolve the lowest resolution to time $t = 10,000M$ to demonstrate long-term stability and evolve two higher resolutions to an earlier time to assess convergence with spatial resolution. *Left:* Analytic initial data is in harmonic coordinates. Higher resolutions evolved to $t = 5,000M$. The highest resolution has $10\times$ tighter time stepper tolerances. *Right:* Analytic initial data is in Kerr-Schild coordinates. Higher resolutions evolved to $t = 2,000M$. The difference between the initial data and evolution gauge causes non-trivial dynamics.

3.3.1.3 Damped Harmonic Coordinates

Our final single-black-hole test consists of evolving Kerr-Schild analytic initial data in damped harmonic gauge with $r_{\text{in}} = r_{\text{exc}} = 1.8M$. The left panel of Fig. 3.5 shows the constraints as a function of time for several different resolutions. Just as in the harmonic gauge case, the non-trivial gauge dynamics cause larger constraint violations until after one light-crossing time to the outer boundary of $r_{\text{out}} = 1,000M$. The evolutions are stable and converge with increasing resolution. We also repeated this evolution but for a black hole with a dimensionless spin of $\chi = 0.8$ and $r_{\text{in}} = r_{\text{exc}} = 0.57M$. We show the constraint violations in the right panel of Fig. 3.5. Again we see stable evolutions and exponential convergence with increasing p -refinement.

3.3.2 Gauge Wave

As a final test of convergence and stability, we evolve analytic initial data consisting of a gauge wave in flat spacetime, a test conceived in [66] as part of a “standard testbed” for NR codes. Physically, the solution is equivalent to flat spacetime, but the chosen coordinates include a sinusoidal traveling wave that introduces time-

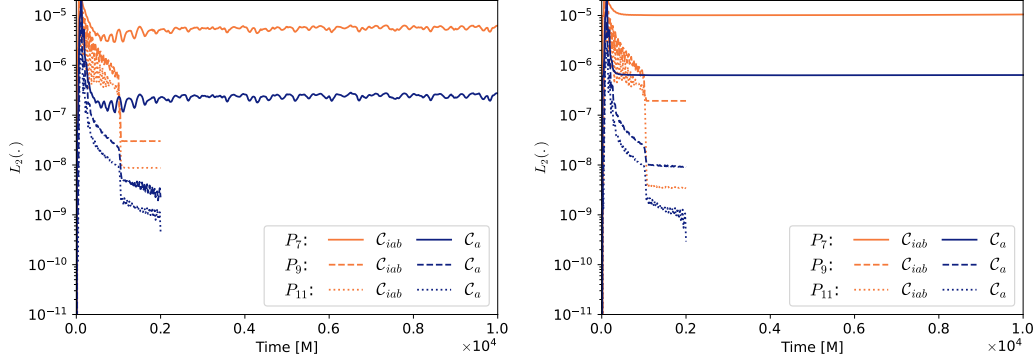


Figure 3.5: Constraint violations for single black hole evolutions in damped harmonic coordinates with Kerr-Schild initial data. We evolve the lowest resolution to time $t = 10,000M$ to demonstrate long-term stability and evolve two higher resolutions to time $t = 2,000M$ to assess convergence with spatial resolution. *Left*: Non-spinning black hole. *Right*: Spinning black-hole with dimensionless spin $\chi = S/M^2 = 0.8$.

dependence, with a line element given by

$$ds^2 = -H(t, x) dt^2 + H(t, x) dx^2 + dy^2 + dz^2, \quad (3.51)$$

where

$$H(t, x) = 1 - A \sin\left(\frac{2\pi(x-t)}{d}\right), \quad (3.52)$$

where A and d are the amplitude and wavelength of the gauge wave, which travels along the x -axis.

We evolve analytic initial data of this solution, using the gauge source function H_a computed directly from the analytic initial data. We set the FOGH constraint damping parameters to $\gamma_0 = \gamma_2 = 1$ and $\gamma_1 = -1$. We evolve on the domain $[0, 1]^3$ with two elements in the x -direction, one element in the y - and z -directions. We fix the y and z points per element to 6 (P_5) and perform a convergence test by running three resolutions with 15 (P_{14}), 18 (P_{17}), and 20 (P_{19}) points per element in the x direction. We apply periodic boundary conditions in all directions. We use a sixth-order Adams-Moulton predictor-corrector time integrator. In our simulations we choose $A = 0.1$ and $d = 1$.

Gauge wave simulations are known to be unstable in the BSSN formulation of the Einstein equations [67], but are stable in the Z4 system [68]. Gauge wave simulations are stable in the FOGH system, as we demonstrate with SpECTRE in Fig. 3.6. The left panel of Fig. 3.6 shows the L_2 norm of the 1-index and 3-index constraints as

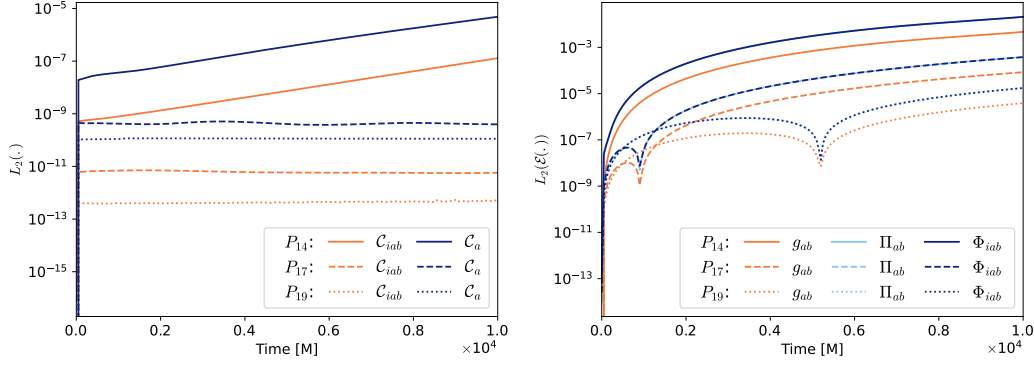


Figure 3.6: Results from an amplitude 0.1 gauge wave simulation demonstrating long-term stable and convergent simulations. We show simulations at three resolutions, 15, 18, and 20 grid points per element in the x -direction. *Left:* The 1-index and 3-index constraints as a function of time. The lowest resolution has exponentially growing constraints in time while the two higher resolutions have constant and convergent constraints in time. *Right:* Difference (error) of the FOGH evolved variables g_{ab} , Π_{ab} , and Φ_{iab} from the analytic solution at the three resolutions. Since these are errors and not constraint violations, we expect the errors will accumulate linearly in time, even for constant constraint violations. We see stable and convergent long-term behavior with the errors decreasing exponentially with increasing resolution. The errors for Π_{ab} and Φ_{iab} lie on top of each other and so only the errors for Φ_{iab} are visible.

a function of time. While the lowest resolution (P_{14}) simulation has exponentially growing constraints, the higher resolution simulations have constant and convergent constraints. Similarly, the right panel of Fig. 3.6 shows the L_2 norm of the error in the evolved variables g_{ab} , Π_{ab} , and Φ_{iab} at the three resolutions. We observe stable and convergent long-term behavior. The highest resolution simulation is close to being limited by the time stepper tolerance.

3.3.3 Binary Black Hole Inspiral, Merger, and Ringdown

In this section, we use SpECTRE to generate binary black hole initial data and evolve the binary through ~ 18 orbits of inspiral, merger, and ringdown to a final, stationary state. We then use SpECTRE’s Cauchy Characteristic Evolution (CCE) module to evolve the outgoing gravitational waves to future null infinity. We perform the simulations at three different resolutions, which we refer to as “Lev0”, “Lev1”, and “Lev2” with Lev0 being the lowest resolution and Lev2 being the highest. Each increase in resolution increases the number of points per element per dimension by one. During the inspiral, each simulation uses 4,800 elements, with Lev0, Lev1, and Lev2 having ~ 2.6 ; ~ 3.7 ; and ~ 5.0 million total grid points. During the

ringdown, each simulation uses 7,680 elements and ~ 10.2 , ~ 13.3 , and ~ 16.9 million total grid points, respectively. All simulations use the damped harmonic gauge to prevent the lapse collapsing and $\sqrt{\gamma}$ diverging at merger. All simulations (both inspiral and ringdown) also use a fourth-order Adams-Moulton predictor-corrector time integrator with absolute and relative time stepper tolerances of 10^{-10} and 10^{-8} , respectively.

The evolutions were each performed on 10 compute-nodes in the Resnick High Performance Computing Center at Caltech. Each compute node has two 28-core Intel Cascade Lake CPUs. Our Lev0, Lev1, and Lev2 evolutions cost 58,000; 71,000; and 117,000 core hours, which amounts to 104; 127; and 209 wallclock hours, and an average of 120; 80; and 41 M /hour during the inspiral. In a future paper, we will assess SpECTRE’s performance and scaling in more detail; our purpose for this paper is to demonstrate that SpECTRE can evolve binary black holes through inspiral, merger, and ringdown.

3.3.3.1 Initial Data

We begin our evolutions with initial data of two equal-mass and non-spinning black holes in a quasicircular orbit. To generate the initial data we use the SpECTRE initial data module [69–71], which solves the elliptic constraint sector of the Einstein equations in the extended conformal thin sandwich (XCTS) formalism [72–74]. It uses the superposed Kerr-Schild formalism to construct a conformal background to the extended conformal thin sandwich equations based on the weighted superposition of two isolated Kerr-Schild black holes [75, 76]. The black holes are represented as excisions with negative-expansion apparent horizon boundary conditions [77, 78]. The initial data solver uses DG methods similar to those described in this article to achieve scalable and parallelizable solutions to the elliptic equations and is also open source [45, 70].

We choose an initial coordinate separation of the excision centers to be $d_0 = 15.366M$ to facilitate future comparison with the family of simulations in [79] and to place the two black holes at ~ 18 orbits before they merge. The masses and spins of the black holes measured on the horizons are driven to the desired values in a control loop that adjusts the initial data parameters, similar to [80]. In a second control loop we performed eccentricity reduction by evolving the initial data for a few orbits and adjusting the initial orbital parameters to iteratively reduce the eccentricity of the orbit [81, 82] to 7.2×10^{-4} . The resulting initial data parameters

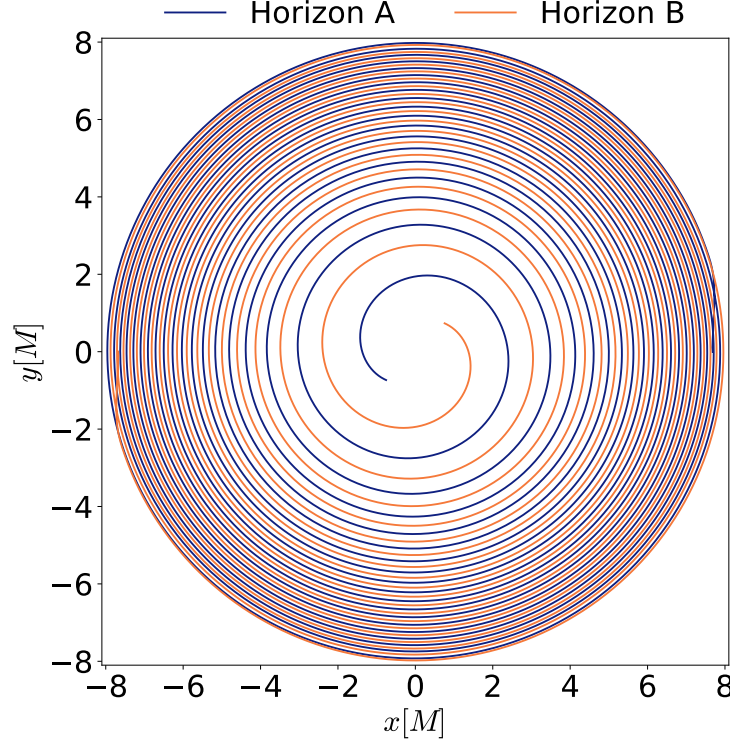


Figure 3.7: The trajectories of the centers of each apparent horizon during the Lev2 inspiral evolution, in the evolution’s asymptotically inertial coordinates. The coordinates x and y indicate the position of the black hole within the orbital plane in units of the total initial mass M of the binary.

Table 3.1: Parameters used to generate the binary-black-hole initial data evolved in § 3.3.3: in terms of the sum of the black holes’ Christodoulou masses $M = M_1 + M_2$ of each black hole, the individual masses M_1 and M_2 , the dimensionless spin magnitudes $|\chi_1|/M^2$ and $|\chi_2|/M^2$, the orbital eccentricity e , the initial coordinate separation d_0 of the horizon centers, the initial orbital angular velocity Ω_0 , and the initial expansion rate $\dot{a}_0$.

M_1/M	M_2/M	$ \chi_1 /M^2$	$ \chi_2 /M^2$	d_0/M	$M\Omega_0$	e	$\dot{a}_0$
0.5	0.5	3×10^{-8}	3×10^{-8}	15.366	0.0159	7.2×10^{-4}	3.4×10^{-5}

are summarized in Table 3.1, and a plot of the resulting inertial trajectories for Lev2 is shown in Fig. 3.7.

3.3.3.2 Computational Domain Decomposition

The numerical evolution of the FOGH system is performed with a DG scheme in which the physical domain of the problem is partitioned into deformed hexahedral

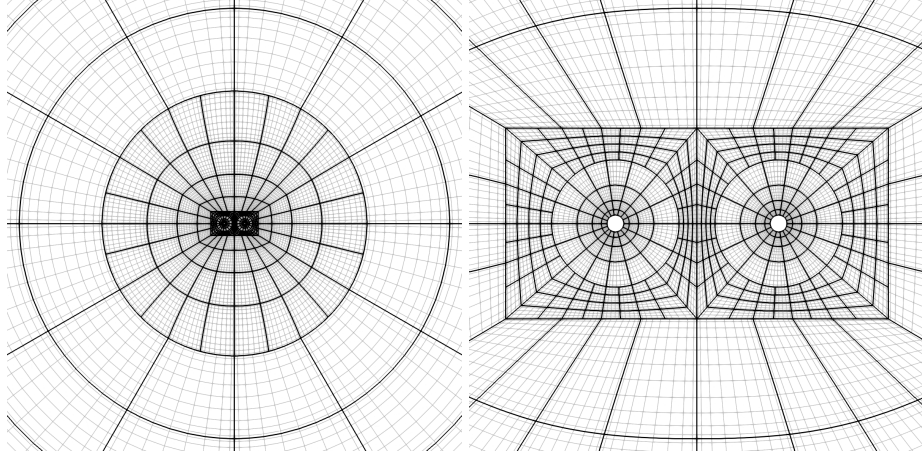


Figure 3.8: An illustration of the computational grid used during the inspiral. We make use of two excision regions, each region lying inside a black hole’s apparent horizon. Each excision is surrounded by a spherical shell partitioned into six deformed cubes as in Fig. 3.2. Each spherical shell is then surrounded by another shell of six deformed cubes that transition to a cubical boundary. Then the two cubes themselves are surrounded by a transitional envelope which becomes spherical. *Left:* The transitional envelope. *Right:* A close-up of the domain structure around the excisions. The center of each excision is offset from the center of the cube.

elements with conforming boundaries. The boundaries and gridpoint distributions of an element are determined by a continuous and differentiable coordinate map applied to the logical Cartesian coordinates, which we label ξ, η and ζ , of a regular cube spanning $[-1, 1]^3$. The maps corresponding to neighboring elements are required to be continuous but are not required to be differentiable at element boundaries. This provides the flexibility necessary to construct the complicated domains needed for binary merger simulations using DG methods. An example of the domain used during the inspiral is shown in Fig. 3.8 and an example of the ringdown domain is shown in Fig. 3.9. While the coordinate maps provide significant flexibility, we found that instabilities arise if neighboring elements differ by significantly more than a factor of two in size, placing a practical constraint on how quickly the resolution can be reduced as one moves away from the BHs.

Our computational domain is the region of space between the outer spherical boundary and the excision boundaries that remove the black hole singularities from the domain. The excision boundaries are spherical in the comoving coordinates with their sizes and shapes in the inertial coordinates informed by the size and shape of the apparent horizons (§ 3.2.7). At merger the apparent horizons of the inspiraling black holes become enveloped by a single common apparent horizon. We handle

the different number of excision boundaries during the inspiral and ringdown by having distinct domains for each. At merger we interpolate data from the inspiral domain that has two excision boundaries to the ringdown domain that has one excision boundary. We describe each domain below.

The inspiral domain:

The inspiral domain is more complicated than the ringdown domain. This is because of the complexity of having two excisions. During the inspiral we must tile a two-excision domain with conforming hexahedra. Our solution to this tiling problem makes use of 44 element collections grouped into two radial and two “biradial” layers. Our description of the domain decomposition starts at the excision surfaces and extends radially outward.

The first layer consists of six wedges composing the spherical shell surrounding each excision. Each wedge is subdivided into multiple elements. The black holes are located on the x -axis at $x \sim \pm 7.683M$ with an excision radius of $r_{\text{exc}} \sim 0.792M$. The shell around each black hole has an outer radius of $6M$. For a fixed target accuracy, distributing the grid points logarithmically in radius and equiangularly [83, 84] in angles significantly reduces computational cost.

The second layer uses a set of wedges that wrap the shells around each black hole in a cubical shell, as seen in Fig. 3.8. A consequence of the decreasing separation between the two black holes during the inspiral is that the size of the excision within each cube grows as the simulation progresses. Since we only deform the region inside the cubes to conform to the shape of the apparent horizons, the simulation will fail if the apparent horizons grow beyond the cube boundaries. We remedy this by decoupling the excision center from the center of the cube (see the right panel of Fig. 3.8), effectively increasing the size of the cube relative to the size of the excision by a constant factor that is sufficient to keep the apparent horizons within the cubes throughout the simulation. This generalized map is crucial for robust inspiral and merger simulations.

The third layer consists of the elements surrounding the cubes around each black hole (layers 1 and 2). We refer to this region, which extends from $r \sim 18M$ to $r = 100M$, as the “envelope”. This layer serves to transition the grid point distributions from what is used near the black holes to the distribution that is used in the wave zone. We use a logarithmic map in the radial direction and we interpolate between a “biradial”

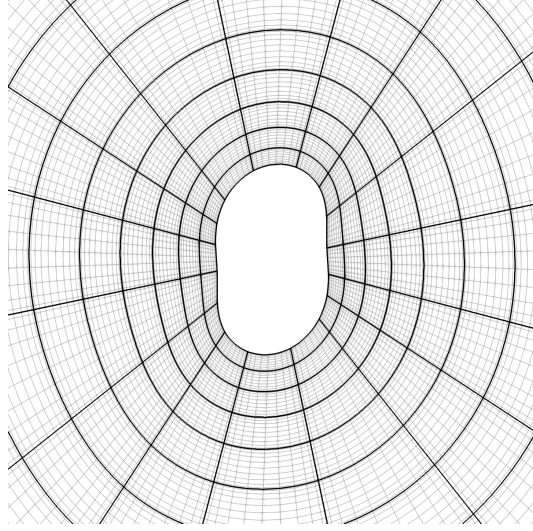


Figure 3.9: An image of the ringdown domain. After a common horizon is found, the evolution is regridded onto a domain with a single excision as in Fig. 3.2. Additional time-dependent maps are used to deform the excision’s spherical shape into one that corresponds to the common horizon shape.

equiangular map used for two excisions and a “radial” equiangular map suited for the spherical outer boundary.

The fourth and final layer is a spherical shell extending from the end of the envelope ($r = 100M$) to the outer boundary at $r = 600M$. This shell uses a linear distribution in the radial direction and an equiangular distribution in the angular directions. Since the GW wavelength is constant in radius, a linear distribution is necessary to avoid under resolving the waves. In production quality simulations we expect to place the outer boundary at $r = 1,500M$, since in typical SpEC simulations we find that this radius is necessary to avoid a center-of-mass drift caused by the gauge boundary condition. Errors in the gauge boundary condition fall off as $\sim 1/r^2$. We find that the drift is larger for longer simulations, but this can be compensated for with a larger outer boundary. Since we use CCE to extract the gravitational waves, a large domain for wave extraction is not required, as it would be if we instead extrapolated the waves to spatial infinity.

In addition to the hexahedral maps that partition the domain, we also globally apply rotation and expansion maps that track the angular, radial, and center of mass motion of the binary system.

To characterize the errors in these coordinate mappings, we introduce the diagnostic quantity (which we refer to as the “Jacobian diagnostic”) as

$$C_i = 1 - \frac{\sum_i |\partial_i x^i|}{\sum_i |D_i x^i|}, \quad (3.53)$$

following the convention in § 3.2.2 where hatted quantities are in the logical coordinates and non-hatted quantities are in the inertial coordinates. In the numerator, $\partial_i x^i$ is the analytic Jacobian provided by the analytic coordinate mappings (both time dependent and time-independent) described in this section. In the denominator, $D_i x^i$ is the Jacobian computed by taking numerical derivatives of the inertial coordinates in each logical direction. The sums are over all gridpoints. If $C_i = 0$, the analytic and numerical Jacobians are identical, meaning that our coordinate mappings are perfectly represented by our Legendre-Gauss-Lobatto DG scheme.

The ringdown domain:

The ringdown domain is a single excision domain used after a common apparent horizon has formed. An example of this domain is shown in Fig. 3.9. It is similar in structure to the domain used in the single black hole tests in § 3.3.1. We use a logarithmic radial map from the excision surface to $r = 50M$ and use a linear spacing further away to resolve the gravitational waves. We used an equidistant map instead of an equiangular map in the angular directions.

A significant challenge compared to the single black hole evolution is that the time-dependent maps used during ringdown must be initialized from, and matched to, the corresponding time-dependent maps in the inspiral. The rotation and expansion maps are matched and then decay exponentially to being time-independent. Most challenging are the shape and size maps. For the shape map we perform a least squares fit in time to the spherical harmonic coefficients of the common horizon (and their time derivatives) found during the inspiral. We fit to 100 times, and then initialize the shape of the excision by evaluating the fit at the transition time. For the size of the excision, we manually specified an excision radius $r_{\text{exc}} = 1.45$ and gave the excision surface an initial outward radial velocity of 1.0; we adjusted these choices by hand until the ringdown was able to begin successfully, with the excision surface remaining inside the common apparent horizon while having all characteristic speeds pointing out of the computational domain (cf. 3.2.3).

For the simulations presented here, we carried out the transition from the inspiral domain to the ringdown domain manually. First, when the coordinate separation of the centers of the black holes' apparent horizons in the asymptotically inertial

frame fell to less than $2.38M$ (a value chosen by hand to roughly correspond to the time that the common horizon first forms), we chose to have the simulation begin outputting the evolution variables at every gridpoint every $\Delta t = 0.01M$. Then, we chose to terminate the inspiral portion of the simulation when the coordinate separation between the two black holes decreased to less than $2.138M$. This value was chosen to be late enough that it yielded enough distinct common horizon finds, but also early enough that the constraint energy (cf. § 3.3.3.4) didn't grow too large. In the future, we intend to implement the techniques that SpEC uses to automate the transition process.

3.3.3.3 Constraint Damping

Based on our experience evolving binary black holes in SpEC, we use a superposition of three Gaussians and a constant for γ_0 and γ_2 , and a single Gaussian plus a constant for γ_1 . See (3.4) for how the constraint damping terms appear in the evolution equations. The motivation for the different Gaussians is to increase the constraint damping near each black hole, which requires the Gaussians to move with the black holes as they inspiral. In SpEC and SpECTRE we achieve this by making the Gaussians functions of the comoving “grid frame” coordinates $x^{\bar{i}}$. As the black holes inspiral, their coordinate radius increases in the comoving $x^{\bar{i}}$ coordinates, which means the width w of the Gaussian must also increase by the same amount. Increasing the width is achieved by dividing the width by the expansion factor $E(t)$, which starts at 1 and decreases as the black holes inspiral. The specific form of the damping parameters we use is

$$\gamma_0(t, x^{\bar{i}}) = \gamma_2(t, x^{\bar{i}}) = C + \sum_{I=0}^2 A_I \exp \left[- \left(\frac{\bar{r}_I}{\bar{w}_I/E(t)} \right)^2 \right], \quad (3.54)$$

$$\gamma_1(t, x^{\bar{i}}) = C + A_0 \exp \left[- \left(\frac{\bar{r}_0}{\bar{w}_0} \right)^2 \right], \quad (3.55)$$

where C is a constant, A_I are the amplitudes of the Gaussians, $\bar{r}_I$ are the grid-frame radii from the center of each Gaussian, and $\bar{w}_I$ are the widths of the Gaussians in the grid frame. Table 3.2 shows the parameters during the inspiral and Table 3.3 shows them during the ringdown. In the grid frame the black holes are always located on the $\bar{x}$ -axis so we only specify the grid frame $\bar{x}$ -coordinate at which each Gaussian is centered, denoted by $\bar{x}^C$ in tables 3.2 and 3.3.

Table 3.2: Parameters for the Gaussians that make up the constraint damping functions during the inspiral. See (3.54) and (3.55) for how the coefficients appear in the functional form of the constraint damping parameters.

	C	A_0	$\bar{w}_0$	$\bar{x}_0^C$	A_1	$\bar{w}_1$	$\bar{x}_1^C$	A_2	$\bar{w}_2$	$\bar{x}_2^C$
γ_0	$0.01/M$	$0.75/M$	38.415	0	$8/M$	3.5	7.683	$8/M$	3.5	-7.683
γ_1	$-0.999/M$	$0.999/M$	$10d_0$	—	—	—	—	—	—	—
γ_2	$0.01/M$	$0.75/M$	38.415	0	$8/M$	3.5	7.683	$8/M$	3.5	-7.683

Table 3.3: Parameters for the Gaussians that make up the constraint damping functions during the ringdown. See (3.54) and (3.55) for how the coefficients appear in the functional form of the constraint damping parameters.

	C	A_0	$\bar{w}_0$	$\bar{x}_0^C$	A_1	$\bar{w}_1$	$\bar{x}_1^C$	A_2	$\bar{w}_2$	$\bar{x}_2^C$
γ_0	$0.01/M$	$1.0/M$	100	0	$7/M$	2.5	0	—	—	—
γ_1	$-0.999999/M$	—	—	—	—	—	—	—	—	—
γ_2	$0.001/M$	$0.1/M$	100	0	$7/M$	2.5	0	—	—	—

3.3.3.4 Constraint Violations

Fig. 3.10 shows the constraint energy $\mathcal{E}$ (see (3.19)) as a function of time at each resolution. Experience from SpEC suggests that the initial, rapid growth in $\mathcal{E}$ is caused by not resolving the initial data’s rapid relaxation and emission of spurious, high-frequency “junk” gravitational radiation at the start of the simulation. Resolving the junk radiation is computationally expensive and generally not done in NR evolutions of binary black holes. After the initial growth of constraint violation damps away, the constraints converge exponentially with increasing p -refinement. The constraints grow sharply near the time of merger, as the black holes become more distorted by each others’ tidal gravity, causing the solution to be less resolved by our fixed computational mesh. We anticipate that future SpECTRE simulations using adaptive mesh refinement will improve the behavior of the constraints near the time of merger.

To ensure the errors from our coordinate mappings do not affect our constraint violations, we compare the L2-norms of the constraint energy and the Jacobian diagnostic in Table 3.4 at $t = 3000M$ during the inspiral. For all our resolutions, the mapping error is below our constraint violations, meaning that our coordinate maps do not affect the results of these simulations.

After the merger, as the remnant black hole rings down, the constraint violations decrease to much smaller values again. When the black hole has settled to its

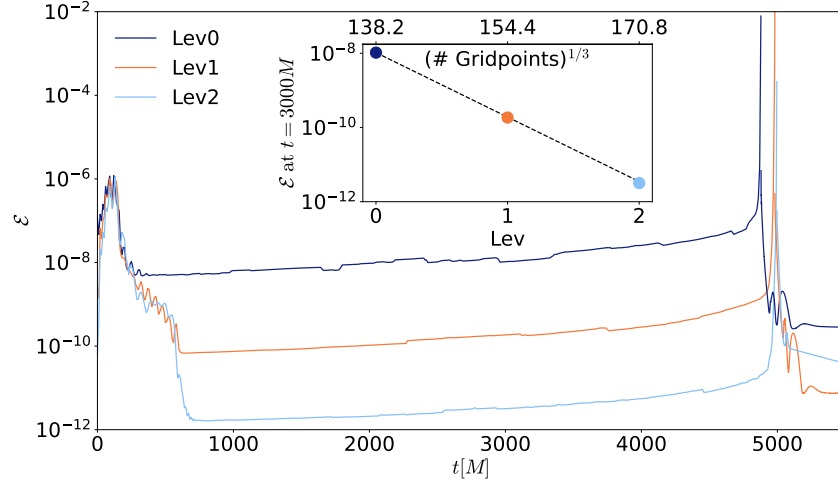


Figure 3.10: L2-norm of the constraint energy $\mathcal{E}$ (see (3.19)) over the entire computational domain for evolutions of an equal-mass, non-spinning, binary black hole at low (“Lev0”), medium (“Lev1”), and high (“Lev2”) spatial resolution, as a function of time t in units of the initial total mass M . The peak of the constraint energy for each resolution is when we transition to the ringdown grid. We see exponential convergence with resolution except at merger and early ringdown, where we suspect adaptive mesh refinement will be necessary to resolve the additional dynamics. The inset shows this exponential convergence at time $t = 3000M$ during the inspiral as a function of the cube root of the total number of grid points (and “Lev”).

Table 3.4: The L2-norm of the constraint energy $\mathcal{E}$ and Jacobian diagnostic C_i over all gridpoints at $t = 3000M$ during the inspiral. For all of our resolutions, the Jacobian diagnostic (which is a measure of our mapping error) is below the error of the constraint energy.

L2-norm	Lev0	Lev1	Lev2
$\mathcal{E}$	1.0×10^{-8}	1.9×10^{-10}	3.2×10^{-12}
C_i	8.8×10^{-11}	6.2×10^{-12}	5.7×10^{-13}

final stationary state, the constraint violations continue to slowly decrease in time. Because the ringdown constraints in the highest resolution are not smaller than those with the medium spatial resolution, we suspect that the numerical error is dominated not by spatial resolution in the ringdown but by some other factor. One possibility is the time-stepping accuracy during the late inspiral and ringdown. We leave a careful study of this, including improvements to the domain decomposition and use of adaptive mesh refinement during the ringdown, to future work.

3.3.3.5 Apparent Horizons

Another way of measuring the accuracy of BBH simulations is to track the masses and spins of the BHs using the apparent horizon surfaces. We find the apparent horizon surfaces using a “fast flow” approach similar to the one outlined in [85]. There are a number of subtleties that arise when implementing this approach with an asynchronous task-based parallelism setting. These are due to the fact that the “fast flow” approach is an iterative method and thus needs to store the metric on the full computational domain at a given simulation time until all iterations are complete. A more in-depth explanation of this will be available in [65].

We measure masses and spins on the individual apparent horizons every $0.5M$ during the inspiral and merger and on the common apparent horizon every $0.1M$ during ringdown. The irreducible mass, defined as $M_{\text{irr}} \equiv \sqrt{A/16\pi}$, where A is the surface area of the apparent horizon, should be monotonically increasing. Thus, any decreases in M_{irr} can be viewed as a measure of the numerical error in the simulation. Another useful metric is the Christodoulou mass $M_{\text{Ch}} \equiv \sqrt{M_{\text{irr}}^2 + S^2/4M_{\text{irr}}^2}$, which includes both the irreducible mass and rotational kinetic energy. The dimensionless spin $\chi = S/M_{\text{Ch}}^2$ measures the spin in terms of approximate rotational Killing vectors, as discussed in Appendix A of [75]. For equal-mass non-spinning simulations M_{irr} and M_{Ch} should remain constant until merger, while χ should remain identically zero. Deviations from this behavior help quantify numerical errors in the simulation.

Fig. 3.11 shows M_{irr} , M_{Ch} , and χ during the inspiral and ringdown. We find that the masses and spins over time are convergent, in the sense that the difference between Lev0 and Lev1 is greater than the difference between Lev1 and Lev2. During the inspiral, after the initial transient, the masses and spins remain more constant in time as resolution increases, until sharp gains near the time of merger as the black holes gain energy and angular momentum. During the ringdown, the masses and

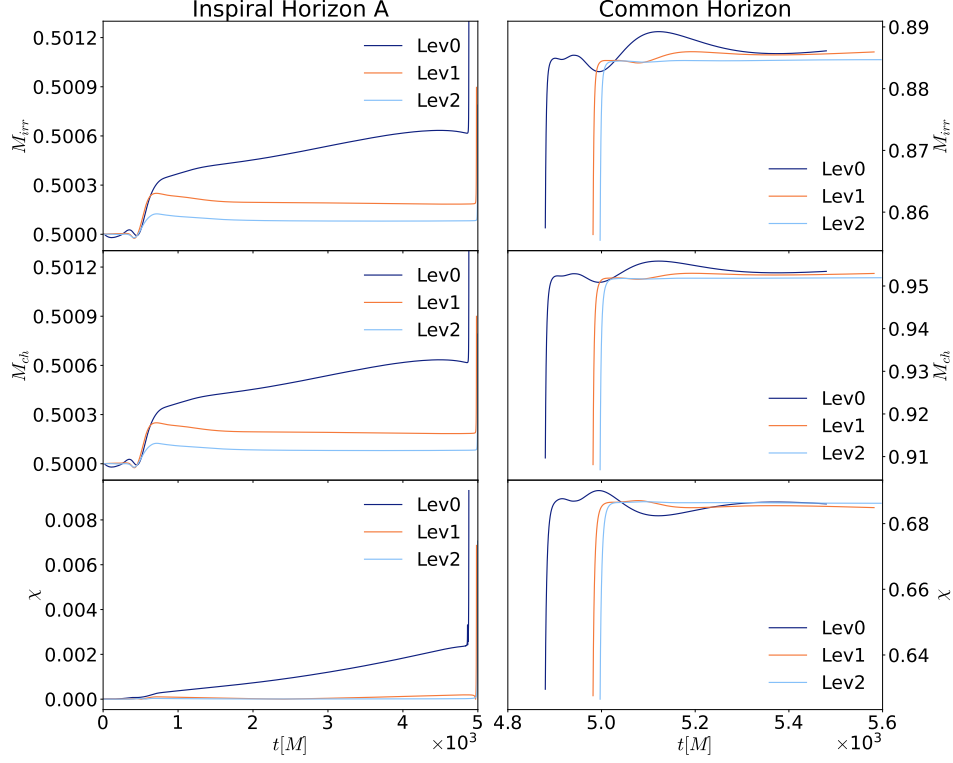


Figure 3.11: Apparent horizon masses and spins as a function of time. The left column shows the irreducible mass M_{irr} , Christodoulou mass M_{ch} , and dimensionless spin angular momentum $\chi \equiv S/M_{\text{ch}}^2$ for one of the individual apparent horizons during inspiral, as a function of time t in units of the total initial Christodoulou mass M . The right column shows the same quantities during the ringdown for the common apparent horizon. We see convergence with increasing resolution, specifically that the Lev1 and Lev2 evolutions track each other more closely than the Lev0 and Lev1 simulations.

spins relax to final, constant values, as expected. The final Christodoulou mass $M_{\text{ch}} = 0.952$ differs from the initial Christodoulou mass ($M_{\text{ch}} = 1$) by 4.8%, and the final spin is 0.686. Both values are consistent with the fitting formulas in [79], tuned using SpEC evolutions of equal-mass, equal-aligned-spin binary black holes.

3.3.3.6 Gravitational Waveforms

We compute gravitational waveforms using Cauchy-Characteristic Evolution (CCE) [46, 86–89], using the SpECTRE implementation of CCE [47]. This method utilizes an additional characteristic evolution code, the one described in [47], that solves the full Einstein equations on a set of outgoing null slices that extend from some inner worldtube all the way to future null infinity. Boundary conditions on the world-

tube are provided by the interior Cauchy evolution, in this case also done with SpECTRE. For the characteristic evolution, there is freedom to choose one complex function on the initial null slice, which encodes the initial incoming radiation. We set it according to equation (16) of [47]. From the characteristic evolution, one can compute the gravitational-wave strain and all five Weyl scalars at future null infinity. Gravitational waveforms computed via CCE are in a well-defined gauge modulo Bondi-van der Burg-Metzner-Sachs (BMS) transformations [90, 91], which are extensions of Poincaré transformations and correspond to symmetries of asymptotically flat spacetimes at future null infinity. The raw output from CCE is in an effectively random BMS frame, so to completely fix the gauge, it is necessary to perform a BMS transformation [92]. We choose to transform waveforms into the superrest frame of the inspiral [93, 94], which can be thought of as the BMS extension of a frame in which the binary is at rest during the inspiral. We find the BMS transformation to map to this frame using data from the strain and Weyl scalars over the time window $[1800M, 2200M]$. See [92] and references therein for an in-depth review of BMS frame fixing.

Fig. 3.12 shows the leading-order $\ell = m = 2$ spin-weighted spherical harmonic mode of the gravitational-wave strain as a function of retarded time t . To estimate the accuracy of the waveforms, we first apply a time shift t_{22}^{peak} so that the peak amplitudes at each spatial resolution occur at the same shifted time. Then, we apply a constant phase offset such that the gravitational-wave phase of the $\ell = m = 2$ mode vanishes at time $t - t_{22}^{\text{peak}} = -4000M$. This time was chosen as an early time after most of the initial spurious “junk” radiation (especially visible in the amplitude at early times) has been emitted. This junk radiation is characteristically different than the junk radiation seen in Fig. 3.10. The junk radiation in Fig. 3.12 is of a lower frequency and is due to our choice of data on the initial null slice. It is an active area of research to improve data on the initial null slice.

We choose to post-process the data in this way because it enables us to transform each simulation to some reasonable BMS frame without using information from the other simulations. As a result, we can perform more meaningful convergence tests, since the output from each simulation is independent of every other. While one could obtain better agreement between different resolutions by finding the BMS transformation which minimizes the residual between the simulations’ waveforms, this would go beyond computing a convergence error, because the frame of one simulation is determined by the other.

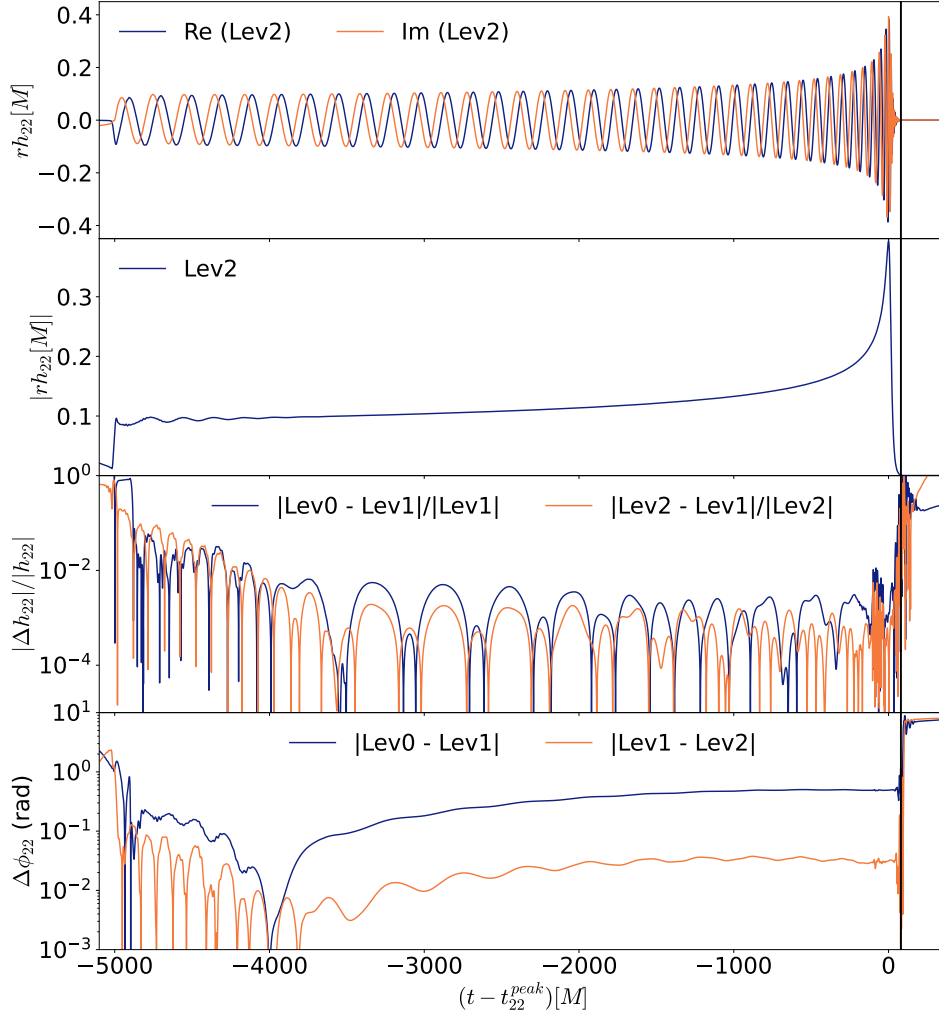


Figure 3.12: The $\ell = m = 2$ spin-weighted spherical-harmonic mode of the gravitational waves, computed using Cauchy-Characteristic Evolution (CCE) and frame-fixed using the Python package `scri`, with a worldtube radius of $200M$, where M is the total initial Christodoulou mass. From top to bottom, as a function of retarded time $t - t_{22}^{\text{peak}}$, the panels show i) the real and imaginary part of the strain rh_{22}/M ; ii) the amplitude $|rh_{22}/M|$ at the highest spatial resolution; iii) the fractional amplitude difference $|\Delta h_{22}|/|h_{22}|$, defined as the magnitude of the $\ell = m = 2$ amplitude difference between two spatial resolutions, divided by the magnitude of the higher spatial resolution's amplitude; iv) the difference in phase $\Delta\phi_{22}$, defined as the difference between the $\ell = m = 2$ gravitational-wave phase at two different spatial resolutions, with each phase offset such that the phase vanishes at $t - t_{22}^{\text{peak}} = -4000M$. The vertical black lines show where the amplitude goes below $\sim 10^{-3}$.

Finally, we interpolate the amplitudes and phases at each spatial resolution onto a common set of times and take differences, estimating the numerical error of a spatial resolution in terms of its difference with the next highest spatial resolution. We find that between time $t - t_{22}^{\text{peak}} = -4000M$ and merger time $t - t_{22}^{\text{peak}} = 0$, the amplitude and phase differences decrease with increasing resolution, as expected, with the medium and high resolution differing in amplitude by 0.01% at time of merger and about 0.1% throughout the inspiral. During the window between $t - t_{22}^{\text{peak}} = -4000M$ and $t = t_{22}^{\text{peak}}$, the medium and high spatial resolutions accumulate 0.03 radians of phase error. After merger time, the fractional amplitude and phase errors grow, which is expected, because the amplitude itself is exponentially falling to zero, making determining the phase accurately increasingly challenging.

3.4 Conclusion

We present the first inspiral-merger-ringdown simulations of two black holes using DG methods. We use the open-source numerical relativity code SpECTRE [45] to perform all simulations. These include several long-term stability and accuracy tests, e.g. evolutions of a single black hole in Kerr-Schild, harmonic, and damped harmonic gauge, as well as a long-term stable gauge wave simulation. All simulations demonstrate the expected exponential convergence.

The binary black hole simulation is of the last 18 orbits before merger of an equal-mass non-spinning binary. We extract gravitational waveforms at future null infinity using SpECTRE's CCE module. We observe exponential convergence in the constraint violations with increasing resolution, and demonstrate convergence in amplitude and phase of the $\ell = m = 2$ mode of the gravitational wave strain. The simulations presented here are the first binary merger simulations where the initial data, evolution, and wave extraction are all performed using the open-source code SpECTRE.

Our medium and high resolution SpECTRE simulations run at 80 and 41 M/hour on ten 56-core Intel Cascade Lake nodes during inspiral. A proper comparison between this performance from SpECTRE and an analogous calculation in SpEC would be nontrivial, in part because a meaningful comparison should ensure that the SpEC and SpECTRE simulations have the same accuracy. We plan to perform such a comparison in the future. Here, for an initial ballpark comparison, we simply note that a SpEC evolution of analogous initial data, at what we typically would consider high resolution in SpEC (approximately 300,000 grid points, with SpEC's

adaptive mesh refinement algorithm varying the precise number of points throughout the simulation), ran at approximately 90M/hour on 32 Intel Sky Lake cores. The SpECTRE high-resolution calculation ran about half as fast in wall time, using about 17 times more cores on a grid with about 17 times more gridpoints. SpECTRE’s domain has many more elements with fewer points per element, enabling it to scale to more CPU cores than SpEC; however, SpEC’s choice to use fewer elements with higher number of points (especially spherical shells near the horizons and in the wave zone) mean that SpEC’s domain is currently much more efficient.

We expect that SpECTRE would outperform SpEC in wall time at sufficiently high resolutions, running on enough CPU cores, but that it would be less efficient, requiring more cost for the same accuracy. We also expect that planned future optimizations (discussed in the next paragraph) will greatly reduce the computational cost of a SpECTRE binary-black-hole calculation at a given accuracy. Finally, note that in its current state, SpECTRE is still efficient enough to perform an ~ 18 orbit inspiral—longer than almost all inspirals published to date using moving-puncture codes—at feasible cost (less than 9 days of wall time on less than 600 compute cores).

While the results here present a milestone for SpECTRE simulations, several further advancements are necessary to enable building catalogs for future gravitational wave detectors. These fall in one of four categories: i) performance improvements like using adaptive mesh refinement, dynamic load balancing, and GPU support; ii) robustness and parameter space improvements like ensuring that high-spin, high-mass-ratio, and eccentric simulations can be performed robustly without hand tuning; iii) automation infrastructure that allows a single user to run hundreds of simulations, such as automatically restarting failed simulations, automatically transitioning from inspiral to ringdown, and automatically running CCE after the Cauchy simulation completes; and iv) improving documentation and tutorials to make the code more accessible to the broader community. Item i) also includes developing a good understanding of how error diagnostics, like constraint violations, impact gravitational-wave amplitude and phase errors; our experience with SpEC suggests that the answer is complicated and will require careful investigation. These will allow SpECTRE to outperform our current code SpEC and to be more useful to the broader numerical-relativity community.

Acknowledgments

Charm++/Converse [95] was developed by the Parallel Programming Laboratory in the Department of Computer Science at the University of Illinois at Urbana-Champaign. The figures in this article were produced with matplotlib [96, 97], TikZ [98], ParaView [99, 100], numpy [101], scipy [102], and scri [103–106]. Computations were performed at the Resnick High-Performance Computing Center; a facility supported by the Resnick Sustainability Institute; at Caltech, the mbot cluster at Cornell, the ocean cluster at Cal State Fullerton, and the Urania cluster at the Max Planck Computing and Data Facility. We are pleased to thank Josh Smith for helpful discussions. This work was supported in part by NSF awards PHY-2208014 and AST-2219109, the Dan Black Family Trust, and Nicholas and Lee Begovich at Cal State Fullerton. This material is based upon work supported by the National Science Foundation under Grants No. PHY-2407742, No. PHY-2207342, and No. OAC-2209655 at Cornell. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was supported by the Sherman Fairchild Foundation at Cornell. Support for this work was provided by NASA through the NASA Hubble Fellowship grant number HST-HF2-51562.001-A awarded by the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Incorporated, under NASA contract NAS5-26555. This work was supported in part by the Sherman Fairchild Foundation and by NSF Grants No. PHY-2309211, No. PHY-2309231, and No. OAC-2209656 at Caltech. P.K. acknowledges support of the Department of Atomic Energy, Government of India, under project no. RTI4001, and by the Ashok and Gita Vaish Early Career Faculty Fellowship at the International Centre for Theoretical Sciences.

Bibliography

- [1] Robert Abbott et al. GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run. *Phys. Rev. X*, 13(4):041039, 2023. doi: 10.1103/PhysRevX.13.041039.
- [2] Luc Blanchet. Post-Newtonian theory for gravitational waves. *Living Rev. Rel.*, 27(1):4, 2024. doi: 10.1007/s41114-024-00050-z.
- [3] Frans Pretorius. Evolution of binary black hole spacetimes. *Phys. Rev. Lett.*, 95:121101, 2005. doi: 10.1103/PhysRevLett.95.121101.

- [4] Manuela Campanelli, Carlos O. Lousto, Pedro Marronetti, and Yosef Zlochower. Accurate evolutions of orbiting black-hole binaries without excision. *Phys. Rev. Lett.*, 96:111101, 2006. doi: 10.1103/PhysRevLett.96.111101.
- [5] John G. Baker, Joan Centrella, Dae-Il Choi, Michael Koppitz, and James van Meter. Gravitational wave extraction from an inspiraling configuration of merging black holes. *Phys. Rev. Lett.*, 96:111102, 2006. doi: 10.1103/PhysRevLett.96.111102.
- [6] Harald P. Pfeiffer. Numerical simulations of compact object binaries. *Class. Quant. Grav.*, 29:124004, 2012. doi: 10.1088/0264-9381/29/12/124004.
- [7] Matthew D. Duez and Yosef Zlochower. Numerical relativity of compact binaries in the 21st century. *Rept. Prog. Phys.*, 82(1):016902, 2019. doi: 10.1088/1361-6633/aadb16.
- [8] Abdul H. Mroue et al. Catalog of 174 binary black hole simulations for gravitational wave astronomy. *Phys. Rev. Lett.*, 111(24):241104, 2013. doi: 10.1103/PhysRevLett.111.241104.
- [9] Karan Jani, James Healy, James A. Clark, Lionel London, Pablo Laguna, and Deirdre Shoemaker. Georgia Tech catalog of gravitational waveforms. *Class. Quant. Grav.*, 33(20):204001, 2016. doi: 10.1088/0264-9381/33/20/204001.
- [10] James Healy, Carlos O. Lousto, Yosef Zlochower, and Manuela Campanelli. The RIT binary black hole simulations catalog. *Class. Quant. Grav.*, 34(22):224001, 2017. doi: 10.1088/1361-6382/aa91b1.
- [11] James Healy, Carlos O. Lousto, Jacob Lange, Richard O’Shaughnessy, Yosef Zlochower, and Manuela Campanelli. Second RIT binary black hole simulations catalog and its application to gravitational waves parameter estimation. *Phys. Rev. D*, 100(2):024021, 2019. doi: 10.1103/PhysRevD.100.024021.
- [12] Michael Boyle et al. The SXS collaboration catalog of binary black hole simulations. *Class. Quant. Grav.*, 36(19):195006, 2019. doi: 10.1088/1361-6382/ab34e2.
- [13] James Healy and Carlos O. Lousto. Third RIT binary black hole simulations catalog. *Phys. Rev. D*, 102(10):104018, 2020. doi: 10.1103/PhysRevD.102.104018.
- [14] James Healy and Carlos O. Lousto. Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits. *Phys. Rev. D*, 105(12):124010, 2022. doi: 10.1103/PhysRevD.105.124010.
- [15] Deborah Ferguson et al. Second MAYA catalog of binary black hole numerical relativity waveforms. *Phys. Rev. D*, 112(4):044043, 2025. doi: 10.1103/gk7x-9cds.

- [16] Michele Punturo et al. The Einstein Telescope: A third-generation gravitational wave observatory. *Class. Quantum Grav.*, 27(19):194002, October 2010. doi: 10.1088/0264-9381/27/19/194002.
- [17] Matthew Evans et al. A horizon study for Cosmic Explorer: Science, observatories, and community. September 2021. URL <https://arxiv.org/abs/2109.09882>.
- [18] Pau others Amaro-Seoane. Laser interferometer space antenna. *arXiv e-prints*, art. arXiv:1702.00786, February 2017. doi: 10.48550/arXiv.1702.00786. URL <https://arxiv.org/abs/1702.00786>.
- [19] Michael Pürrer and Carl-Johan Haster. Gravitational waveform accuracy requirements for future ground-based detectors. *Phys. Rev. Res.*, 2(2):023151, 2020. doi: 10.1103/PhysRevResearch.2.023151.
- [20] Deborah Ferguson, Karan Jani, Pablo Laguna, and Deirdre Shoemaker. Assessing the readiness of numerical relativity for LISA and 3G detectors. *Phys. Rev. D*, 104(4):044037, 2021. doi: 10.1103/PhysRevD.104.044037.
- [21] Aasim Jan, Deborah Ferguson, Jacob Lange, Deirdre Shoemaker, and Aaron Zimmerman. Accuracy limitations of existing numerical relativity waveforms on the data analysis of current and future ground-based detectors. *Phys. Rev. D*, 110(2):024023, 2024. doi: 10.1103/PhysRevD.110.024023.
- [22] Hengrui Zhu, Jacob Fields, Francesco Zappa, David Radice, James M. Stone, Alireza Rashti, William Cook, Sebastiano Bernuzzi, and Boris Daszuta. Performance-portable numerical relativity with AthenaK. *Astrophys. J. Suppl.*, 278(2):50, 2025. doi: 10.3847/1538-4365/adcf96.
- [23] <https://www.black-holes.org/SpEC.html>.
- [24] Jan S. Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge University Press, Cambridge, UK, 2007.
- [25] Alireza Rashti, Francesco Maria Fabbri, Bernd Brügmann, Swami Vivekanandji Chaurasia, Tim Dietrich, Maximiliano Ujevic, and Wolfgang Tichy. New pseudospectral code for the construction of initial data. *Phys. Rev. D*, 105(10):104027, 2022. doi: 10.1103/PhysRevD.105.104027.
- [26] L. Jens Papenfort, Samuel D. Tootle, Philippe Grandclément, Elias R. Most, and Luciano Rezzolla. New public code for initial data of unequal-mass, spinning compact-object binaries. *Phys. Rev. D*, 104(2):024057, 2021. doi: 10.1103/PhysRevD.104.024057.
- [27] Hannes R. Rüter, David Hilditch, Marcus Bugner, and Bernd Brügmann. Hyperbolic relaxation method for elliptic equations. *Phys. Rev. D*, 98(8):084044, 2018. doi: 10.1103/PhysRevD.98.084044.

- [28] Wolfgang Tichy, Liwei Ji, Ananya Adhikari, Alireza Rashti, and Michal Pirog. The new discontinuous Galerkin methods based numerical relativity program Nmesh. *Class. Quant. Grav.*, 40(2):025004, 2023. doi: 10.1088/1361-6382/aca7.
- [29] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. A pseudospectral method for gravitational wave collapse. *Phys. Rev. D*, 93(6):063006, 2016. doi: 10.1103/PhysRevD.93.063006.
- [30] Marcus Bugner, Tim Dietrich, Sebastiano Bernuzzi, Andreas Weyhausen, and Bernd Brügmann. Solving 3D relativistic hydrodynamical problems with weighted essentially nonoscillatory discontinuous Galerkin methods. *Phys. Rev. D*, 94(8):084004, 2016. doi: 10.1103/PhysRevD.94.084004.
- [31] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. Evolutions of centered Brill waves with a pseudospectral method. *Phys. Rev. D*, 96(10):104051, 2017. doi: 10.1103/PhysRevD.96.104051.
- [32] Isabel Suárez Fernández, Rodrigo Vicente, and David Hilditch. Semilinear wave model for critical collapse. *Phys. Rev. D*, 103(4):044016, 2021. doi: 10.1103/PhysRevD.103.044016.
- [33] Maitraya K. Bhattacharyya, David Hilditch, K. Rajesh Nayak, Sarah Renkhoff, Hannes R. Rüter, and Bernd Brügmann. Implementation of the dual foliation generalized harmonic gauge formulation with application to spherical black hole excision. *Phys. Rev. D*, 103(6):064072, 2021. doi: 10.1103/PhysRevD.103.064072.
- [34] Isabel Suárez Fernández, Sarah Renkhoff, Daniela Cors, Bernd Brügmann, and David Hilditch. Evolution of Brill waves with an adaptive pseudospectral method. *Phys. Rev. D*, 106(2):024036, 2022. doi: 10.1103/PhysRevD.106.024036.
- [35] Sarah Renkhoff, Daniela Cors, David Hilditch, and Bernd Brügmann. Adaptive hp refinement for spectral elements in numerical relativity. *Phys. Rev. D*, 107(10):104043, 2023. doi: 10.1103/PhysRevD.107.104043.
- [36] Thomas W. Baumgarte, Bernd Brügmann, Daniela Cors, Carsten Gundlach, David Hilditch, Anton Khirnov, Tomáš Ledvinka, Sarah Renkhoff, and Isabel Suárez Fernández. Critical phenomena in the collapse of gravitational waves. *Phys. Rev. Lett.*, 131(18):181401, 2023. doi: 10.1103/PhysRevLett.131.181401.
- [37] Daniela Cors, Sarah Renkhoff, Hannes R. Rüter, David Hilditch, and Bernd Brügmann. Formulation improvements for critical collapse simulations. *Phys. Rev. D*, 108(12):124021, 2023. doi: 10.1103/PhysRevD.108.124021.

- [38] Krinio Marouda, Daniela Cors, Hannes R. Rüter, Florian Atteneder, and David Hilditch. Twist-free axisymmetric critical collapse of a complex scalar field. *Phys. Rev. D*, 109(12):124042, 2024. doi: 10.1103/PhysRevD.109.124042.
- [39] Florian Atteneder, Hannes R. Rüter, Daniela Cors, Roxana Rosca-Mead, David Hilditch, and Bernd Brügmann. Boson star head-on collisions with constraint-violating and constraint-satisfying initial data. *Phys. Rev. D*, 109(4):044058, 2024. doi: 10.1103/PhysRevD.109.044058.
- [40] Michael Dumbser, Olindo Zanotti, and Ilya Peshkov. High order discontinuous Galerkin schemes with subcell finite volume limiter and AMR for a monolithic first-order BSSNOK formulation of the Einstein–Euler equations. *arXiv e-prints*, art. arXiv:2406.15798, June 2024. doi: 10.48550/arXiv.2406.15798. URL <https://arxiv.org/abs/2406.15798>.
- [41] Miguel Alcubierre, Werner Bengert, Bernd Bruegmann, Gerd Lanfermann, Lars Nierger, Edward Seidel, and Ryoji Takahashi. The 3-d grazing collision of two black holes. *Phys. Rev. Lett.*, 87:271103, 2001. doi: 10.1103/PhysRevLett.87.271103.
- [42] William H Reed and TR Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Lab., N. Mex.(USA), 1973. URL <https://www.osti.gov/biblio/4491151>.
- [43] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer-Verlag New York, New York, 2008. doi: 10.1007/978-0-387-72067-8.
- [44] Béla Szilágyi, Jonathan Blackman, Alessandra Buonanno, Andrea Taracchini, Harald P. Pfeiffer, Mark A. Scheel, Tony Chu, Lawrence E. Kidder, and Yi Pan. Approaching the post-Newtonian regime with numerical relativity: A compact-object binary simulation spanning 350 gravitational-wave cycles. *Phys. Rev. Lett.*, 115(3):031102, 2015. doi: 10.1103/PhysRevLett.115.031102.
- [45] Nils Deppe, William Throwe, Lawrence E. Kidder, Nils L. Vu, Kyle C. Nelli, Cristóbal Armaza, Marceline S. Bonilla, François Hébert, Yoonsoo Kim, Prayush Kumar, Geoffrey Lovelace, Alexandra Macedo, Jordan Moxon, Eamonn O’Shea, Harald P. Pfeiffer, Mark A. Scheel, Saul A. Teukolsky, Nikolas A. Wittek, et al. *SpECTRE v2024.09.29*. 10.5281/zenodo.13858965, September 2024. URL <https://spectre-code.org>.
- [46] Jordan Moxon, Mark A. Scheel, and Saul A. Teukolsky. Improved Cauchy-characteristic evolution system for high-precision numerical relativity waveforms. *Phys. Rev. D*, 102(4):044052, August 2020. doi: 10.1103/PhysRevD.102.044052.

- [47] Jordan Moxon, Mark A. Scheel, Saul A. Teukolsky, Nils Deppe, Nils Fischer, Francois Hébert, Lawrence E. Kidder, and William Throwe. SpECTRE Cauchy-characteristic evolution system for rapid, precise waveform extraction. *Phys. Rev. D*, 107(6):064013, 2023. doi: 10.1103/PhysRevD.107.064013.
- [48] Thomas W. Baumgarte and Stuart L. Shapiro. *Numerical Relativity: Solving Einstein's Equations on the Computer*. Cambridge University Press, 2010. doi: 10.1017/CBO9781139193344.
- [49] L. Rezzolla and O. Zanotti. *Relativistic Hydrodynamics*. Oxford University Press, September 2013.
- [50] Lee Lindblom, Mark A. Scheel, Lawrence E. Kidder, Robert Owen, and Oliver Rinne. A new generalized harmonic evolution system. *Class. Quant. Grav.*, 23:S447–S462, 2006. doi: 10.1088/0264-9381/23/16/S09.
- [51] Lee Lindblom, Keith D. Matthews, Oliver Rinne, and Mark A. Scheel. Gauge drivers for the generalized harmonic Einstein equations. *Phys. Rev. D*, 77:084001, 2008. doi: 10.1103/PhysRevD.77.084001.
- [52] Bela Szilagyi, Lee Lindblom, and Mark A. Scheel. Simulations of binary black hole mergers using spectral methods. *Phys. Rev. D*, 80:124010, 2009. doi: 10.1103/PhysRevD.80.124010.
- [53] Saul A. Teukolsky. Formulation of discontinuous Galerkin methods for relativistic astrophysics. *J. Comput. Phys.*, 312:333–356, 2016. doi: 10.1016/j.jcp.2016.02.031.
- [54] Nils Deppe, François Hébert, Lawrence E. Kidder, and Saul A. Teukolsky. A high-order shock capturing discontinuous Galerkin–finite difference hybrid method for GRMHD. *Class. Quant. Grav.*, 39(19):195001, 2022. doi: 10.1088/1361-6382/ac8864.
- [55] Mark A. Scheel, Harald P. Pfeiffer, Lee Lindblom, Lawrence E. Kidder, Oliver Rinne, and Saul A. Teukolsky. Solving Einstein's equations with dual coordinate frames. *Phys. Rev.*, D74:104006, 2006. doi: 10.1103/PhysRevD.74.104006.
- [56] Jan S. Hesthaven. A stable penalty method for the compressible Navier–Stokes equations: II. One-dimensional domain decomposition schemes. *SIAM J. Sci. Comput.*, 18:658–685, 1997.
- [57] Jan S. Hesthaven. A stable penalty method for the compressible Navier–Stokes equations: III. Multidimensional domain decomposition schemes. *SIAM J. Sci. Comput.*, 20:62–93, 1999.
- [58] Jan S. Hesthaven. Spectral penalty methods. *Appl. Num. Math.*, 33:23–41, 2000.

- [59] Oliver Rinne, Lee Lindblom, and Mark A. Scheel. Testing outer boundary treatments for the Einstein equations. *Class. Quant. Grav.*, 24:4053–4078, 2007. doi: 10.1088/0264-9381/24/16/006.
- [60] Sizheng Ma et al. Fully relativistic three-dimensional Cauchy-characteristic matching for physical degrees of freedom. *Phys. Rev. D*, 109(12):124027, 2024. doi: 10.1103/PhysRevD.109.124027.
- [61] Luisa T. Buchman, Matthew D. Duez, Marlo Morales, Mark A. Scheel, Tim M. Kosterstiz, and Andrew M. Evans. Numerical relativity multi-modal waveforms using absorbing boundary conditions. *arXiv e-prints*, art. arXiv:2402.12544, February 2024. doi: 10.48550/arXiv.2402.12544. URL <https://arxiv.org/abs/2402.12544>.
- [62] William Throwe and Saul Teukolsky. Local time-stepping with predictor-corrector methods. In preparation.
- [63] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, September 2007. URL <http://numerical.recipes>.
- [64] Daniel A. Hemberger, Mark A. Scheel, Lawrence E. Kidder, Béla Szilágyi, Geoffrey Lovelace, Nicholas W. Taylor, and Saul A. Teukolsky. Dynamical excision boundaries in spectral evolutions of binary black hole spacetimes. *Class. Quant. Grav.*, 30:115001, 2013. doi: 10.1088/0264-9381/30/11/115001.
- [65] Kyle C. Nelli, William Throwe, Nils Deppe, Mark A. Scheel, Lawrence E. Kidder, Nils L. Vu, and Saul A. Teukolsky. Horizon tracking for asynchronous parallel black hole simulations. August 2025. URL <https://arxiv.org/abs/2508.08408>.
- [66] Miguel Alcubierre et al. Toward standard testbeds for numerical relativity. *Class. Quant. Grav.*, 21(2):589–613, 2004. doi: 10.1088/0264-9381/21/2/019.
- [67] Maria C. Babiuc et al. Implementation of standard testbeds for numerical relativity. *Class. Quant. Grav.*, 25:125012, 2008. doi: 10.1088/0264-9381/25/12/125012.
- [68] Daniela Alic, Carles Bona-Casas, Carles Bona, Luciano Rezzolla, and Carlos Palenzuela. Conformal and covariant formulation of the Z4 system with constraint-violation damping. *Phys. Rev. D*, 85:064040, 2012. doi: 10.1103/PhysRevD.85.064040.
- [69] Nils L. Fischer and Harald P. Pfeiffer. Unified discontinuous Galerkin scheme for a large class of elliptic equations. *Phys. Rev. D*, 105(2):024034, 2022. doi: 10.1103/PhysRevD.105.024034.

- [70] Nils L. Vu et al. A scalable elliptic solver with task-based parallelism for the spectre numerical relativity code. *Phys. Rev. D*, 105(8):084027, 2022. doi: 10.1103/PhysRevD.105.084027.
- [71] Nils L. Vu. Discontinuous Galerkin scheme for elliptic equations on extremely stretched grids. *Phys. Rev. D*, 110(8):084062, 2024. doi: 10.1103/PhysRevD.110.084062.
- [72] James W. York, Jr. Conformal 'thin sandwich' data for the initial-value problem of general relativity. *Phys. Rev. Lett.*, 82:1350–1353, 1999. doi: 10.1103/PhysRevLett.82.1350.
- [73] Harald P. Pfeiffer and James W. York, Jr. Extrinsic curvature and the Einstein constraints. *Phys. Rev. D*, 67:044022, 2003. doi: 10.1103/PhysRevD.67.044022.
- [74] Harald P. Pfeiffer. The initial value problem in numerical relativity. *J. Hyperbolic Differ. Equ.*, 2(02):497–520, 2005. doi: 10.1142/S0219891605000518. URL <https://doi.org/10.1142/S0219891605000518>.
- [75] Geoffrey Lovelace, Robert Owen, Harald P. Pfeiffer, and Tony Chu. Binary-black-hole initial data with nearly-extremal spins. *Phys. Rev. D*, 78:084017, 2008. doi: 10.1103/PhysRevD.78.084017.
- [76] Geoffrey Lovelace, Mark A. Scheel, and Bela Szilagyi. Simulating merging binary black holes with nearly extremal spins. *Phys. Rev. D*, 83:024010, 2011. doi: 10.1103/PhysRevD.83.024010.
- [77] Gregory B. Cook and Harald P. Pfeiffer. Excision boundary conditions for black hole initial data. *Phys. Rev. D*, 70:104016, 2004. doi: 10.1103/PhysRevD.70.104016.
- [78] Vijay Varma, Mark A. Scheel, and Harald P. Pfeiffer. Comparison of binary black hole initial data sets. *Phys. Rev. D*, 98(10):104011, 2018. doi: 10.1103/PhysRevD.98.104011.
- [79] Daniel A. Hemberger, Geoffrey Lovelace, Thomas J. Loredo, Lawrence E. Kidder, Mark A. Scheel, Béla Szilágyi, Nicholas W. Taylor, and Saul A. Teukolsky. Final spin and radiated energy in numerical simulations of binary black holes with equal masses and equal, aligned or anti-aligned spins. *Phys. Rev. D*, 88:064014, 2013. doi: 10.1103/PhysRevD.88.064014.
- [80] Serguei Ossokine, Francois Foucart, Harald P. Pfeiffer, Michael Boyle, and Béla Szilágyi. Improvements to the construction of binary black hole initial data. *Class. Quantum Gravity*, 32:245010, 2015. doi: 10.1088/0264-9381/32/24/245010.

- [81] Alessandra Buonanno, Lawrence E. Kidder, Abdul H. Mroué, Harald P. Pfeiffer, and Andrea Taracchini. Reducing orbital eccentricity of precessing black-hole binaries. *Phys. Rev. D*, 83:104034, May 2011. doi: 10.1103/PhysRevD.83.104034. URL <https://link.aps.org/doi/10.1103/PhysRevD.83.104034>.
- [82] Sarah Habib, Mark A. Scheel, and Saul A. Teukolsky. Eccentricity reduction for quasicircular binary evolutions. *Phys. Rev. D*, 111(8):084059, 2025. doi: 10.1103/PhysRevD.111.084059.
- [83] Robert Sadourny. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Wea. Rev.*, 100:136–144, 1972.
- [84] Corrado Ronchi, Roberto Iacono, and Pier Stanislao Paolucci. The “cubed sphere”: A new method for the solution of partial differential equations in spherical geometry. *Journal of Computational Physics*, 124(1): 93–114, 1996. ISSN 0021-9991. doi: <https://doi.org/10.1006/jcph.1996.0047>. URL <https://www.sciencedirect.com/science/article/pii/S0021999196900479>.
- [85] Carsten Gundlach. Pseudospectral apparent horizon finders: An efficient new algorithm. *Phys. Rev. D*, 57:863–875, 1998. doi: 10.1103/PhysRevD.57.863.
- [86] Nigel T. Bishop, Roberto Gómez, Luis Lehner, and Jeffrey Winicour. Cauchy-characteristic extraction in numerical relativity. *Phys. Rev. D*, 54(10):6153–6165, November 1996. doi: 10.1103/PhysRevD.54.6153.
- [87] Nigel T. Bishop and Luciano Rezzolla. Extraction of gravitational waves in numerical relativity. *Living Rev. Relativ.*, 19(1):2, October 2016. doi: 10.1007/s41114-016-0001-9.
- [88] Casey J. Handmer, Béla Szilágyi, and Jeffrey Winicour. Spectral Cauchy characteristic extraction of strain, news and gravitational radiation flux. *Class. Quantum Grav.*, 33(22):225007, November 2016. doi: 10.1088/0264-9381/33/22/225007.
- [89] Kevin Barkett, Jordan Moxon, Mark A. Scheel, and Béla Szilágyi. Spectral Cauchy-characteristic extraction of the gravitational wave news function. *Phys. Rev. D*, 102(2):024004, July 2020. doi: 10.1103/PhysRevD.102.024004.
- [90] Hermann Bondi, M. G. J. Van der Burg, and A. W. K. Metzner. Gravitational waves in general relativity, VII. Waves from axi-symmetric isolated system. *Proc. R. Soc. A*, 269(1336):21–52, August 1962. ISSN 2053-9169. doi: 10.1098/rspa.1962.0161. URL <http://dx.doi.org/10.1098/rspa.1962.0161>.

- [91] Rainer Sachs. Gravitational waves in general relativity. VIII. Waves in asymptotically flat space-time. *Proc. R. Soc. A*, 270(1340):103–126, October 1962. ISSN 2053-9169. doi: 10.1098/rspa.1962.0206. URL <http://dx.doi.org/10.1098/rspa.1962.0206>.
- [92] Keefe Mitman et al. A review of gravitational memory and BMS frame fixing in numerical relativity. *Class. Quant. Grav.*, 41(22):223001, 2024. doi: 10.1088/1361-6382/ad83c2.
- [93] Keefe Mitman et al. Fixing the BMS frame of numerical relativity waveforms. *Phys. Rev. D*, 104(2):024051, 2021. doi: 10.1103/PhysRevD.104.024051.
- [94] Keefe Mitman et al. Fixing the BMS frame of numerical relativity waveforms with BMS charges. *Phys. Rev. D*, 106(8):084029, 2022. doi: 10.1103/PhysRevD.106.084029.
- [95] Laxmikant Kale et al. Uiuc-ppl/Charm: Charm++ version 6.10.2. 10.5281/zenodo.3972617, August 2020. URL <https://doi.org/10.5281/zenodo.3972617>.
- [96] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
- [97] Thomas A Caswell, Michael Droettboom, Antony Lee, John Hunter, Eric Firing, Elliott Sales de Andrade, Tim Hoffmann, David Stansby, Jody Klymak, Nelle Varoquaux, Jens Hedegaard Nielsen, Benjamin Root, Ryan May, Phil Elson, Darren Dale, Jae-Joon Lee, Jouni K. Seppänen, Damon McDougall, Andrew Straw, Paul Hobson, Christoph Gohlke, Tony S Yu, Eric Ma, Adrien F. Vincent, Steven Silvester, Charlie Moad, Nikita Kniazev, Hannah, Elan Ernest, and Paul Ivanov. matplotlib/matplotlib: REL: V3.3.0. 10.5281/zenodo.3948793, July 2020. URL <https://doi.org/10.5281/zenodo.3948793>.
- [98] Till Tantau. The TikZ and PGF packages. github.com/pgf-tikz/pgf. URL <https://github.com/pgf-tikz/pgf>.
- [99] James Ahrens, Berk Geveci, and Charles Law. ParaView: An end-user tool for large-data visualization. In *Visualization Handbook*. Butterworth-Heinemann, Burlington, 2005. ISBN 978-0123875822. doi: 10.1016/B978-012387582-2/50038-1. URL <https://doi.org/10.1016/B978-012387582-2/50038-1>.
- [100] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., Clifton Park, NY, USA, 2015. ISBN 1930934300.
- [101] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H.

- van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [102] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.
- [103] Michael Boyle. Angular velocity of gravitational radiation from precessing binaries and the corotating frame. *Phys. Rev. D*, 87(10):104006, 2013. doi: 10.1103/PhysRevD.87.104006.
- [104] Michael Boyle, Lawrence E. Kidder, Serguei Ossokine, and Harald P. Pfeiffer. Gravitational-wave modes from precessing black-hole binaries. *arXiv e-prints*, art. arXiv:1409.4431, September 2014. doi: 10.48550/arXiv.1409.4431. URL <https://arxiv.org/abs/1409.4431>.
- [105] Michael Boyle. Transformations of asymptotic gravitational-wave data. *Phys. Rev. D*, 93(8):084031, 2016. doi: 10.1103/PhysRevD.93.084031.
- [106] Mike Boyle, Dante Iozzo, and Leo C. Stein. `moble/scri`: V1.2. 10.5281/zenodo.4041972, September 2020. URL <https://doi.org/10.5281/zenodo.4041972>.

Chapter 4

INFRASTRUCTURE FOR CROSS-CODE COMPARISON OF CCE WAVEFORMS

4.1 Introduction

Being able to compare gravitational waveforms with each other, whether it be comparing Numerical Relativity (NR) waveforms with other NR waveforms or observed gravitational waves (GWs), is an extremely important ability in GW science, potentially the most important. GW detectors compare suspected signals to models that have been informed by NR such as the IMRPhenom family (examples: [1–3]) for frequency-domain analysis and the SEOBNR family (examples: [4–6]) for time-domain analysis. NR codes also compare waveforms internally, typically for verification of convergence as in § 3.3.3.6 or for analysis of trends in GW catalogs [7].

However, the coordinate freedom of GR makes this a complicated task. Comparing waveforms from different NR simulations, even from the same code, has to be done carefully since different simulations will be done in different gauges, and therefore many quantities that depend on the gauge can't be directly compared.

4.1.1 Electricity and Magnetism Example

To give an example of why we need to consider gauges when comparing NR waveforms, let us first look at a more familiar use of gauges in an introductory setting such as electricity and magnetism to gain intuition. Maxwell's equations (in vector form) are

$$\nabla \cdot \mathbf{E} = 4\pi\rho \quad \nabla \times \mathbf{E} = -\frac{1}{c} \frac{\partial \mathbf{B}}{\partial t} \quad (4.1)$$

$$\nabla \cdot \mathbf{B} = 0 \quad \nabla \times \mathbf{B} = \frac{1}{c} \left(4\pi\mathbf{J} + \frac{\partial \mathbf{E}}{\partial t} \right) \quad (4.2)$$

From Gauss's law for magnetism and Faraday's equation along with vector calculus identities, we can define

$$\mathbf{B} = \nabla \times \mathbf{A} \quad (4.3)$$

$$\mathbf{E} = -\nabla\phi - \frac{1}{c} \frac{\partial \mathbf{A}}{\partial t} \quad (4.4)$$

where ϕ is the electric potential and $\mathbf{A}$ is the vector potential. We notice that if we modify ϕ and $\mathbf{A}$ according to

$$\mathbf{A} \rightarrow \mathbf{A} + \nabla\psi \quad (4.5)$$

$$\phi \rightarrow \phi - \frac{1}{c} \frac{\partial\psi}{\partial t} \quad (4.6)$$

for some scalar function ψ , $\mathbf{B}$ and $\mathbf{E}$ in Eqs. 4.3 and 4.4 remain unchanged. This ability to choose ϕ and $\mathbf{A}$ is called gauge freedom, and the choice that we make is called the gauge.

In electricity and magnetism there are two common gauges: Coulomb and Lorenz. The Coulomb gauge is defined by

$$\nabla \cdot \mathbf{A} = 0 \quad (4.7)$$

To describe the Lorenz gauge, we combine ϕ and $\mathbf{A}$ as a four-vector

$$A^\mu = (\phi/c, \mathbf{A}). \quad (4.8)$$

The Lorenz gauge is then

$$\partial_\mu A^\mu = 0. \quad (4.9)$$

This differs from the Coulomb gauge because of the $\partial_0 A^0 \sim \partial_t \phi$ part. The implications that this choice of gauge has on the equations can be found in virtually any electricity and magnetism textbook [8, 9]. The important point for this discussion is that even though $\mathbf{E}$ and $\mathbf{B}$ mathematically remain unchanged depending on which gauge is chosen, the gauge dependent quantities ϕ and $\mathbf{A}$ cannot be compared between gauges.

4.1.2 General and Numerical Relativity Gauges

Now we move to gauges in general relativity (GR). Instead of putting a constraint on the four-potential as our gauge condition, in GR we impose our gauge condition on the coordinates x^μ of our spacetime. This coordinate freedom is briefly explored in Ch. 1, but a full derivation can be found in virtually any GR text book, including [10–12] to name a few. For our purposes, let's take a look at a few gauges in GR and NR.

4.1.2.1 Transverse-Traceless Gauge

The first gauge most students encounter is one called the transverse-traceless (TT) gauge, occasionally called the radiation gauge. The TT gauge is particularly useful

in a linearized theory of gravity where we define our metric as

$$g_{\mu\nu} = \eta_{\mu\nu} + \bar{h}_{\mu\nu} \quad (4.10)$$

with

$$\bar{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h \quad (4.11)$$

where $\eta_{\mu\nu}$ is our flat background spacetime metric and $h_{\mu\nu}$ is a small perturbation of this background. Here, $\bar{h}_{\mu\nu}$ is a “trace-reversed” form. We take our field equations

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi T_{\mu\nu} \quad (4.12)$$

and substitute Eq. 4.10 into them. This will contain second derivatives of $\bar{h}_{\mu\nu}$ because of the Ricci tensor and scalar. It is at this point where we make a gauge choice of

$$\square x^\mu = 0 \quad (4.13)$$

which can be shown to simplify to

$$\square x^\mu = 0 \rightarrow g^{\mu\nu}\Gamma_{\mu\nu}^\rho = 0 \rightarrow \partial_\mu \bar{h}_\lambda^\mu = 0 \quad (4.14)$$

in the weak-field (linear) limit[12]. Plugging Eq. 4.14 into Eq. 4.12 and taking the vacuum case ($R_{\mu\nu} = T_{\mu\nu} = 0$), we arrive at a clean result

$$\square \bar{h}_{\mu\nu} = 0. \quad (4.15)$$

It can be shown (through some math that is unimportant to the point here) [12] that

$$h_{\mu\nu}^{\text{TT}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (4.16)$$

where we have dropped the bars because $h_{\mu\nu}$ is the trace-reverse of $\bar{h}_{\mu\nu}$ which is traceless. These are the familiar GW polarizations h_+ and $h_\times$ for a linear GW propagating in the z-direction. Therefore, this is a useful gauge to analyze GWs from one of the detectors (though it’s quite a bit more complicated than this in reality [13–16]).

4.1.2.2 NR Gauges

Moving on to NR, the two most popular formulations of the field equations (Eq. 4.12) are the Generalized Harmonic (GH) formulation [17, 18] and the BSSN/Z4 family [19–25]. Each of these formulations use different gauge conditions for the evolution of the coordinates x^μ , none that are equal to the TT gauge.

We start with GH. There are two dominant gauges used when evolving the GH formulation. The first is actually the harmonic gauge introduced in Eq. 4.13. The TT gauge is the specialization of the harmonic gauge to the linear approximation of GR. However, GH describes the fully non-linear theory of gravity and so the TT gauge is not suitable here. The harmonic gauge is convenient because the solutions for the equations of the coordinates are harmonic (or wave) solutions, hence the name, and we know how to solve for waves.

The second gauge commonly used in GH evolutions is one that actually builds off the harmonic gauge, because its wave solutions are so desirable. The general form of this gauge, called the damped-harmonic gauge, is

$$\square x_\mu = H_\mu(x^\nu, g_{\nu\kappa}) \quad (4.17)$$

where H_μ is called the gauge-source function [26], a function of the coordinates and the spacetime metric. The gauge source function is defined in Eqs. 3.45–3.48. The main purpose for this is to damp away oscillations in the lapse α and drive it to one (so the time evolution remains steady).

For the BSSN/Z4 family of formulations, a gauge condition called the moving-puncture gauge, combined with $1 + \log$ slicing and a Γ -driver condition, is used. For this family, conditions are put onto the lapse function α and the shift vector β^i instead of having equations for the coordinates themselves. The $1 + \log$ slicing condition is (Ch. 4 of [27])

$$\partial_t \alpha = -2\alpha K + \beta^j \partial_j \alpha \quad (4.18)$$

where K is the extrinsic curvature. The Γ -driver condition can come in two forms (Ch. 4 of [27])

$$\partial_t \beta^i = \eta B^i + \beta^j \partial_j \beta^i, \quad (4.19)$$

or ([24])

$$\partial_t \beta^i = \tilde{\Gamma}^i - \eta \beta^i + \beta^j \partial_j \beta^i, \quad (4.20)$$

where η is a tunable parameter, and $\tilde{\Gamma}^i$ is related to the Christoffel symbols. The former is typically used with BSSN while the latter is used with the Z4 family, but many NR codes make slight modifications to any of these gauge equations.

It should be fairly clear that trying to compare a GW produced with the GH formulation to a GW produced with the BSSN formulation is not well defined because the gauges that are used are completely different. However, more subtly, comparing GWs that were produced with the same formulation is also ill-defined. This is due to the limitations of numerical resolution and floating-point arithmetic when solving Einstein's Equations on the computer. Because of these numerical issues, even simulations that start with the same initial data but are evolved with different spatial or temporal resolutions may not evolve in an identical manner. Therefore, the coordinates of these different resolution simulations will be different at the same “time” of the simulation. This can be seen in the bottom two panels of Fig. 3.12 where we are comparing the phase and amplitude difference of the GWs from different resolutions of BBH simulations evolved with the same initial data.

4.2 Cauchy-Characteristic Extraction

Now that we have detailed a number of different useful gauges, namely the TT, harmonic, damped harmonic, and moving-puncture gauges, it should be clear that the GW strains h we compute from simulations done in these different gauges cannot directly be compared with each other. The coordinate freedom of GR implies that even if we use the same formulas to compute h , we will get different answers because the coordinates are completely different. Therefore, how can we compare waveforms across codes? The answer is by comparing the waveforms at a location where there is a well-defined way of choosing coordinates. This location is future null infinity, also denoted $\mathcal{I}^+$. Future null infinity is the location in spacetime that future null rays (e.g. light rays) escape if they don't terminate inside the black hole and aren't in a stable orbit.

There are a few ways of computing GWs at $\mathcal{I}^+$ that allow for comparison of GWs between resolutions or between codes. Most NR codes don't have a computational domain that extends out to $\mathcal{I}^+$, as this would be intractably computationally expensive with typical methods. Instead, GWs at $\mathcal{I}^+$ must be computed in another manner. The first, and by far most inaccurate, method is to simply ignore gauge effects and compare GWs that are extracted at finite radii within the computational domain, treating them as if they were at $\mathcal{I}^+$. While this method cannot be used to

compare waveforms between NR codes, when comparing similar simulations from the same code, it can be a quick-and-dirty method for a rapid analysis.

The second, and by far most common, method for computing GWs at $\mathcal{I}^+$ is called extrapolation. The details for extrapolation can be found in the SXS Collaboration's second catalog release [28]. However, the overall method is to compute the Ψ_4 Weyl scalar and the strain h , either directly using Regge-Wheeler-Zerilli methods [29–31] or taking two time integrals of Ψ_4 , on a number of concentric spheres surrounding the origin of the coordinate system. These quantities are then extrapolated with a $1/r^N$ power law, where N is chosen based on a number of different conditions described in [28].

Extrapolation has been very successful and has been widely used by the NR community for producing GW catalogs [28, 32–34]. However, because it simply extrapolates waveforms from the near field region around the black hole, some non-linear effects of GR as the wave travels to $\mathcal{I}^+$ are not accounted for and may be important. Even with memory-corrected waveforms from [7], they still don't account for all non-linear effects. For this, a full evolution of Einstein's equations out to $\mathcal{I}^+$ is needed.

4.2.1 Characteristic Evolution

A method to evolve Einstein's equations all the way out to $\mathcal{I}^+$ is called Cauchy-Characteristic Extraction/Evolution (CCE). Here, we detail the important concepts and equations of CCE that will be needed in the rest of the chapter, following [35] and [36].

CCE works by first performing a Cauchy evolution of the spacetime using one of the formulations we discussed earlier in this chapter (GH, BSSN, Z4). Then, we store the spacetime metric and its derivatives on a sphere centered at the origin for all times during our simulation. This is called the worldtube. This worldtube data is then used as the inner boundary condition for the Characteristic evolution, which is performed on a domain starting at the worldtube and extending out to $\mathcal{I}^+$, represented with compactified coordinates. This characteristic evolution is performed on null hypersurfaces, unlike the Cauchy evolution which is performed on spacelike hypersurfaces. This is because with the characteristic evolution we are primarily concerned with evolving and resolving GWs, which travel along null rays, making null hypersurfaces well suited to this problem.

CCE decomposes the metric into yet another set of coordinates and gauge conditions,

distinct from those used in GH or BSSN. The metric in a Bondi-like spherical coordinate system $\{u, r, x^A\}$ is

$$ds^2 = - \left(e^{2\beta} \frac{V}{r} - r^2 h_{AB} U^A U^B \right) du^2 - 2e^{2\beta} du dr - 2r^2 h_{AB} U^B du dx^A + r^2 h_{AB} dx^A dx^B \quad (4.21)$$

where h_{AB} is related to the metric on the 2-sphere q_{AB} , V and β are scalar functions, and U^A is a vector on the 2-sphere. Note that β should not be confused with the shift vector β^i . These Bondi-like coordinates have the gauge conditions that $g_{rr} = g_{rA} = 0$, $\det(h_{AB}) = \det(q_{AB})$, and asymptotic behavior

$$\lim_{x \rightarrow \infty} \{\beta, U^A, h_{AB}, \frac{V - r}{r^2}\} = \mathcal{O}(r^0). \quad (4.22)$$

We can define a complex dyad q^A (basis vector) of q_{AB} as follows

$$q_{AB} = \frac{1}{2} (q_A \bar{q}_B + \bar{q}_A q_B), \quad (4.23)$$

with barred quantities being the complex conjugate. Additionally, if a quantity transforms like

$$q_A \rightarrow q_A e^{is\psi} \quad (4.24)$$

for some phase ψ , then we say that quantity has spin weight s . With these definitions, it is now convenient to define some scalars that will be important for the CCE evolution:

$$U \equiv U^A q_A \quad (s = 1) \quad (4.25)$$

$$Q \equiv r^2 e^{-2\beta} q^A h_{AB} \partial_r U^B \quad (s = 1) \quad (4.26)$$

$$r^2 W \equiv V - r \quad (s = 0) \quad (4.27)$$

$$J \equiv \frac{1}{2} q^A q^B h_{AB} \quad (s = 2) \quad (4.28)$$

$$K \equiv \sqrt{1 + J\bar{J}} \quad (s = 0) \quad (4.29)$$

$$H \equiv \partial_u J \quad (s = 2) \quad (4.30)$$

Now we can define a hierarchical set of evolution equations for these scalars which will give us the characteristic evolution of the spacetime on null slices. The first of these equations that is needed is actually Eq. 4.30, which will give us J . Then once we have J , we can build up a solution by solving radial differential equations for the other scalars. Here we will denote the right hand side of each equation with

a generic function S because the details of what is on the right hand side are not important. See [35] again for the full equations.

$$\partial_r \beta = S_\beta(J), \quad (4.31)$$

$$\partial_r(r^2 Q) = S_Q(J, \beta), \quad (4.32)$$

$$\partial_r U = S_U(J, \beta, Q), \quad (4.33)$$

$$\partial_r(r^2 W) = S_W(J, \beta, Q, U), \quad (4.34)$$

$$\begin{aligned} \partial_r(rH) + L_H(J, \beta, Q, U, W)H \\ + L_{\bar{H}}(J, \beta, Q, U, W)\bar{H} = S_H(J, \beta, Q, U, W) \end{aligned} \quad (4.35)$$

The final radial equation for H lets us solve Eq. 4.30 for J at the next time step and the hierarchy starts over again.

Now these scalars are defined in terms of our metric expressed in a Bondi-like gauge which is useful for evolving the characteristic system. However, in order for us to impose boundary conditions with worldtube data from a Cauchy evolution performed in some arbitrary evolved gauge, we must transform the evolved gauge into our Bondi-like gauge. The mathematics of this transformation are quite tedious and unimportant for the point being made here. It uses a number of intermediate coordinates that are only helpful for the transformation, so we will skip them. To see them in detail, refer to [35]. Instead we just state that our scalars defined in Eqs. 4.25-4.30 can be expressed as some function of the Cauchy spacetime metric and Cauchy evolved coordinates (e.g. $U = f(x^\mu, g_{\mu\nu})$, where x^μ is not related to x^A), with all the coordinate transformations hidden.

4.2.2 Frame Fixing and BMS Transformations

Now that we have a way to evolve GWs out to $\mathcal{I}^+$, unfortunately, we still aren't finished. While we could directly compare GWs evolved from CCE, we actually still have some coordinate freedom left to exploit, specifically because we are at $\mathcal{I}^+$. This coordinate freedom is the Bondi-van der Burg-Metzner-Sachs (BMS) group of symmetries, which is the standard Lorentz group plus an infinite-dimensional group called supertranslations [37, 38]. A supertranslation is an angle-dependent time translation where every point on the 2-sphere of $\mathcal{I}^+$ can be time translated independently of each other because they are all causally disconnected. With this freedom, we can take two NR waveforms and map them to a common BMS frame for comparison. The most common frames used are the post-Newtonian (PN) BMS frame (i.e. the frame that PN waveforms are in), the inspiral superrest frame (i.e.

the frame of the inspiraling BHs), and the remnant superrest frame (i.e. the frame of the remnant black hole after a merger) [39]. The transformation of CCE waveforms into one of these frames is called frame fixing.

The math behind frame fixing to the PN BMS or superrest frame is particularly tedious. However, all calculations are provided by the Python code `scri` [40], which accepts the output from SpECTRE CCE. Therefore, in order to compare waveforms one must not only perform characteristic evolutions with SpECTRE CCE, but one must also perform frame fixing on the CCE output of two waveforms being compared to put them into the same BMS frame. While this final frame fixing is crucial for being able to compare CCE waveforms, we only note it here and refer the reader to the sources cited in this section for further details—they are not important to the transformation of Cauchy worldtube data described below.

4.3 Worldtube Data Formats

We have described the basics of characteristic evolution. Specifically, we use the Bondi scalars defined in Eqs. 4.25-4.30 as part of a hierarchical evolution system, we need those Bondi scalars on the worldtube that serves as the inner boundary condition for the characteristic evolution, and these Bondi scalars on the worldtube can be computed from the Cauchy metric, coordinates, and spatial and temporal derivatives of the metric. Next we focus on how this last step is implemented in SpECTRE and what new features were added in SpECTRE.

4.3.1 Original Worldtube Data Format

The first detail we need to discuss is how SpECTRE actually represents and stores the Bondi scalars either in memory of the simulation or on disk. Since we need the Bondi scalars on a sphere of constant radius at each timestep, representing the scalars as complex spherical harmonic coefficients is the most natural. For a surface $S_{\ell,m}$, we require $m = \ell$. This implies we will use $M = (\ell + 1)^2$ complex-valued coefficients (or modes) to represent each Bondi scalar. We will call this the “modal” representation. The modal representation is extremely useful for two reasons. First, it is straightforward to convert between two representations with different ℓ since the conversion is either truncating the number of coefficients or appending coefficients with value zero. Second, the modal representation is significantly more compact than a “nodal” representation where we represent the complex valued scalar at collocation points on a sphere. For the same surface $S_{\ell,m}$ the number of nodes needed to represent the scalar is $N = (\ell + 1)(2\ell + 1) > M$. The advantage with

using a nodal representation for the actual characteristic evolution is that we can easily perform nonlinear arithmetic between Bondi scalars which is needed for the spectral algorithms that SpECTRE uses. This is significantly more difficult with a modal representation because you cannot perform such arithmetic with coefficients.

We now have a situation where it is ideal to represent the evolved Bondi scalars with nodal values, but the worldtube Bondi variables with modal coefficients. The worldtube variables must be written to disk during the Cauchy evolution, so we want those to take up as little disk space as possible. Also, there's no guarantee that the ℓ we write the worldtube variables with will be the same ℓ that we use for the inner boundary of our characteristic evolution. One may even want to run multiple characteristic evolutions with varying resolution.

The SpECTRE CCE algorithm [36] was previously designed to accept two different forms of the Bondi variables on the worldtube. The first is reading in α , β^i , g_{ij} , and their derivatives each written out in complex modal form. This was to support a legacy output format from the SpEC code [41]. The second is directly reading in the Bondi scalars in complex modal form in the Cauchy evolution gauge. This format is output from the modern SpEC simulations [7], and from all SpECTRE simulations [42]. Both SpEC and SpECTRE have routines within the Cauchy evolution that compute the necessary Bondi scalars in the evolution gauge on the worldtube. The SpECTRE CCE algorithm then reads in both these data formats. For the metric variables, a conversion to Bondi scalars is first computed within the CCE algorithm.

4.3.2 New Worldtube Data Formats

In this work, we have added three more input formats for SpECTRE CCE. They are 1) Bondi scalars in complex nodal format, 2) metric variables in complex modal format but without needing the time derivatives of the metric quantities, and 3) metric variables in complex nodal format, both with and without the time derivatives of the metric quantities. We have added support for reading in nodal data formats because it is often easier for other NR codes besides SpEC and SpECTRE to output data onto a set of grid points rather than transforming the data to the exact form of complex spherical harmonic coefficients that SpECTRE requires. Our spherical harmonic formulation, however, does require the worldtube data to be output at specific grid, or collocation, points necessary for the CCE algorithm. To aid in giving the correct collocation points, SpECTRE offers a separate executable that will write the correct collocation points for a given radius to a text file in the necessary

order. SpECTRE also offers a C routine that can be added to the source code that will give an array of the necessary collocation points.

We have expanded support for reading in metric quantities for the worldtube because most other NR codes use either the BSSN or Z4 formulations of Einstein's equations. In particular, we have relaxed the requirement to provide time derivatives of the metric quantities for the worldtube data. This was done because unlike the GH formulation where one of the evolved variables, Π , is the time derivative of the spacetime metric, the BSSN and Z4 formulations only evolve the metric itself. Even though the time derivatives of the metric quantities are computed for the Cauchy evolution, they are often not available for output. The time derivatives within the evolution would be out-of-sync with the actual evolved metric quantities by one time step and trying to sync up the data is a much larger change for other codes.

Therefore, we allow for computing the time derivatives internally in SpECTRE CCE. For the spatial metric we use the 3+1 equation

$$\partial_t \gamma_{ij} = -2\alpha K_{ij} + D_i \beta_j + D_j \beta_i. \quad (4.36)$$

Extreme care must be taken so that we use the correct gauge equations for α and β^i . Luckily for α , most other codes use the $1 + \log$ slicing condition given by Eq. 4.18. Some codes choose to not include the advective term $\beta^j \partial_j \alpha$, so we have added it as optional. For the gamma-driver condition, we support both Eqs. 4.19 and 4.20. Again, we make the advective term $\beta^j \partial_j \beta^i$ optional. For the different gamma-driver conditions, they require different variables to be written to disk from the Cauchy evolution. This is the responsibility of the user to know which variables need to be written out.

Due to the increased number of supported worldtube data formats, the inputs to the actual SpECTRE CCE algorithm have been simplified. We now restrict the worldtube input data format to only be the Bondi variables represented in complex modal coefficients. This was done largely to avoid waiting for extra computations during the characteristic evolution for the worldtube data. It was also done to simplify the code-paths taken during the characteristic evolution. Therefore, in order to support all these worldtube data formats, SpECTRE now offers a separate executable that will transform the worldtube data from any of the supported input formats to the Bondi scalars in complex modal form in a separate output file. The new output file is then fed directly into the CCE algorithm. The new executable

has the ability to handle overlapping output times and multiple output files from different segments of a simulation.

Finally, in order to make these new improvements available to the NR community, SpECTRE now offers a downloadable tar file which has the following pre-built executables:

1. The CCE executable itself plus an associated input file.
2. A preprocessing executable to convert the worldtube data into the Bondi modal format plus associated input files with examples for each of the accepted input data types.
3. An executable to write the collocation points for a sphere at a given radius and angular resolution ℓ to interpolate input data to.
4. Testing scripts plus a small amount of example worldtube data used to verify the executables work.

Additionally, we have added detailed documentation to the SpECTRE webpage with instructions for how to use these executables, recommended settings for certain input parameters, and other small pieces of knowledge that were empirically determined while running CCE.

4.4 Examples of NR Codes Using SpECTRE CCE

Now that we have detailed the process for converting Cauchy worldtube data into the proper format for SpECTRE CCE, we present some examples where this is being used.

4.4.1 SpECTRE

The first example is from § 3.3.3.6. Here we are comparing waveforms from different resolutions of a GH BBH simulation with mass ratio $q = 1$ and non-spinning black holes. Since these simulations were run with SpECTRE, the worldtube data is in the Bondi modal format directly accepted by CCE. The goal of comparing the different resolutions is to assess the convergence of SpECTRE BBH simulations. Therefore, for that study, we did not actually transform the different resolutions to a common superrest frame for comparison. Instead, we transformed each resolution separately to the superrest frame of the inspiral of that resolution (i.e. a frame where the binary is at rest during early inspiral). Doing this allows for a convergence test because

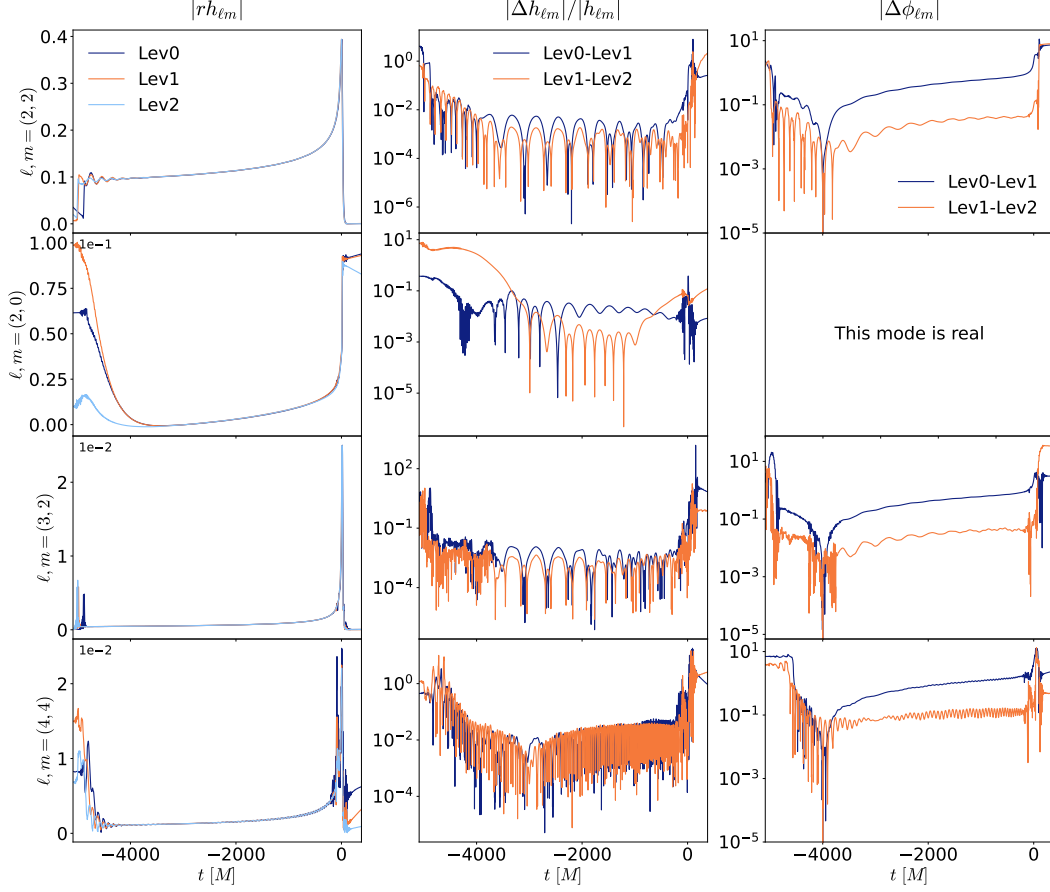


Figure 4.1: SpECTRE GWs mapped to the inspiral superrest frame of the “Lev2” simulation at time $t - t_{22}^{\text{peak}} = -3000M$ before the peak of the amplitude of the 2, 2 mode of the waveform. All waveforms are plotted so $t = 0$ corresponds to this peak of the 2, 2 mode. The four rows are the $\ell, m = (2, 2), (2, 0), (3, 2), (4, 4)$ modes of the waveforms, respectively. The left column is the amplitude of each mode. The middle column is the fractional amplitude difference with respect to one higher Lev. The third column is the phase difference of the waveforms where we zero out the phase of each waveform at $t - t_{22}^{\text{peak}} = -4000M$, consistent with Fig. 3.12. Since the $\ell, m = (2, 0)$ isn’t oscillatory (i.e. real), we omit the phase difference plot for this mode.

each resolution is transformed into a superrest frame without information from the other resolutions, effectively keeping each resolution isolated from the others.

In Fig. 4.1 we choose a common superrest frame, specifically the inspiral superrest frame of the “Lev2” simulation at a time of $t - t_{22}^{\text{peak}} = -3000M$, and then map the other two Levs to this superrest frame. The amplitude and phase differences for the $\ell, m = (2, 2)$ mode are similar to that of Fig. 3.12 and we don’t see a noticeable improvement mapping to the same superrest frame. For the 2, 0 mode, the early

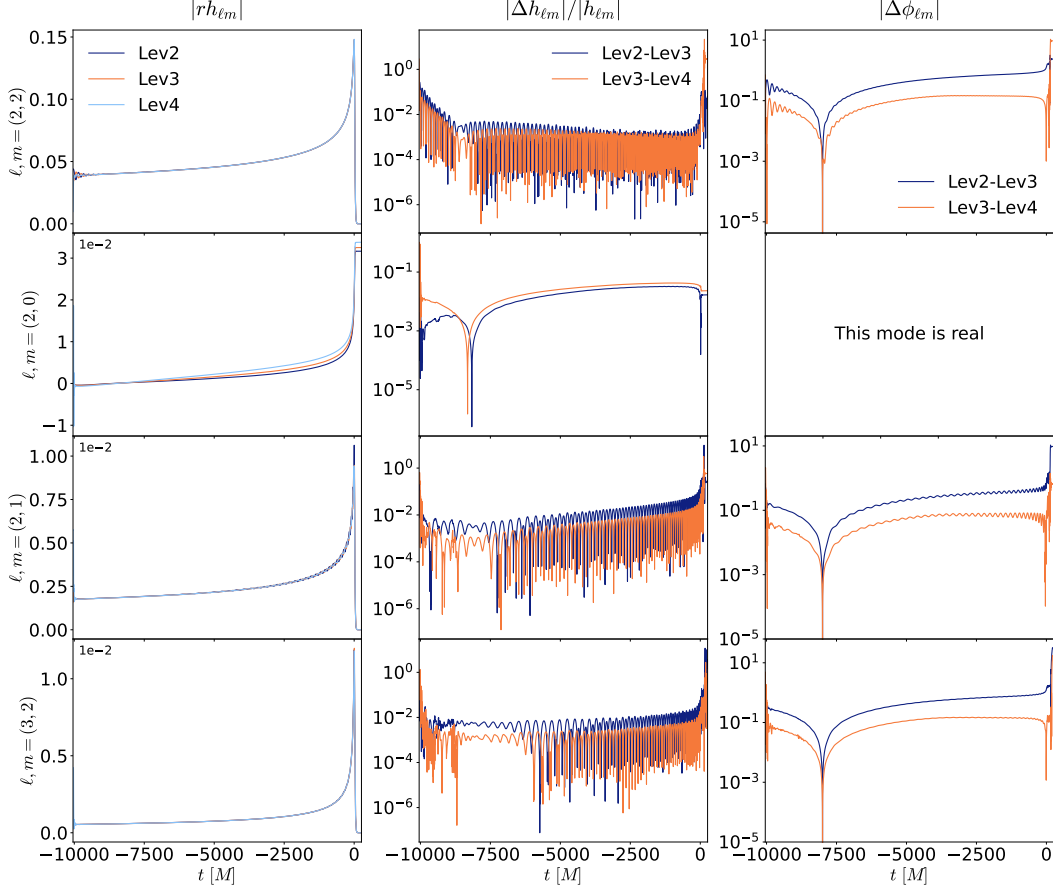


Figure 4.2: SXS:BBH:2696 GWs mapped to the inspiral superrest frame of each Lev at time $t - t_{22}^{\text{peak}} = -8694M$ before the peak of the amplitude of the 2, 2 mode of the waveform. All waveforms are plotted so $t = 0$ corresponds to this peak of the 2, 2 mode. The four rows are the $\ell, m = (2, 2), (2, 0), (2, 1), (3, 2)$ modes of the waveforms, respectively. The left column is the amplitude of each mode. The middle column is the fractional amplitude difference with respect to one higher Lev. The third column is the phase difference of the waveforms where we zero out the phase of each waveform at $t - t_{22}^{\text{peak}} = -8000M$. Since the $\ell, m = (2, 0)$ isn't oscillatory (i.e. real), we omit the phase difference plot for this mode.

time amplitude differences are likely due to poor initial data on the initial null slice of the characteristic evolution. The late time behavior of the 2, 0 mode is likely due to the cubed spheres that SpECTRE uses in the ringdown domain. The 3, 2 and 4, 4 modes show similar behavior to the 2, 2, just at a smaller amplitude.

4.4.2 Spec

The next example is running SpECTRE CCE over a waveform from the SXS catalog [7]. We chose SXS:BBH:2696, a $q = 8$ quasicircular BBH with dimensionless

spin $\vec{\chi} = (0, 0, 0.8)$ on each black hole, which was performed using SpEC and was simulated for 43 orbits. This simulation was also evolved in the GH gauge and output its data in the Bondi modal format directly accepted by CCE. In Fig. 4.2, we show similar plots to Fig. 4.1 with the $(2, 2)$, $(2, 0)$, and $(3, 2)$ modes, but we plot the $(2, 1)$ mode instead of the $(4, 4)$ mode since this simulation is asymmetric in its masses, so we expect the $(2, 1)$ mode to be excited.

In Fig. 4.2, we map each Lev to its own superrest frame at time $t - t_{22}^{\text{peak}} = -8694M$ and then compare the waveforms. All the amplitude and phase comparisons appear to be convergent, except for the fractional amplitude comparison of the $2, 0$ mode. But that only appears slightly non-convergent. Otherwise, the amplitude of the $2, 0$ mode looks significantly better than SpECTRE's. Also, the phase differences in Fig. 4.2 appear to be much better behaved than in SpECTRE, even with a waveform that is roughly double its length.

4.4.3 AthenaK

Next we summarize results from [43] which runs SpECTRE CCE on simulations performed with the AthenaK NR code [44]. AthenaK evolves the Z4c formalism of Einstein's equations using the moving-puncture gauge with the $1 + \log$ condition and the Γ -driver condition, both including the advective terms. In Fig. 4.3 we show an AthenaK waveform run through CCE and frame fixed to a superrest frame at a time just after junk radiation has dissipated. Here we plot the $\ell, m = (2, 2), (2, 0), (2, 1), (3, 2)$ modes of the waveform, again showing that CCE is able to run on worldtube data from other codes. For a more in-depth analysis of this waveform and GW event GW150914, see [43]. In order for AthenaK to compute radial derivatives, metric data was dumped on a number of concentric spheres at radii spaced using Chebyshev collocation points [45]. Radial derivatives are then computed with Chebyshev polynomials around the middle spherical shell [45].

4.4.4 FIL

SpECTRE CCE can run on more than just data from a BBH. Any compact merger with dynamical gravity that can be simulated can supply worldtube data and then be run through CCE. In Fig. 4.4, we present a case of two binary neutron star (BNS) simulations run through CCE. Even though the equations of the characteristic evolution were derived assuming a vacuum spacetime, so long as the amount of matter at the worldtube radius is negligible and does not meaningfully contribute to the curvature of spacetime, we are safe to apply CCE. The two BNS simulations

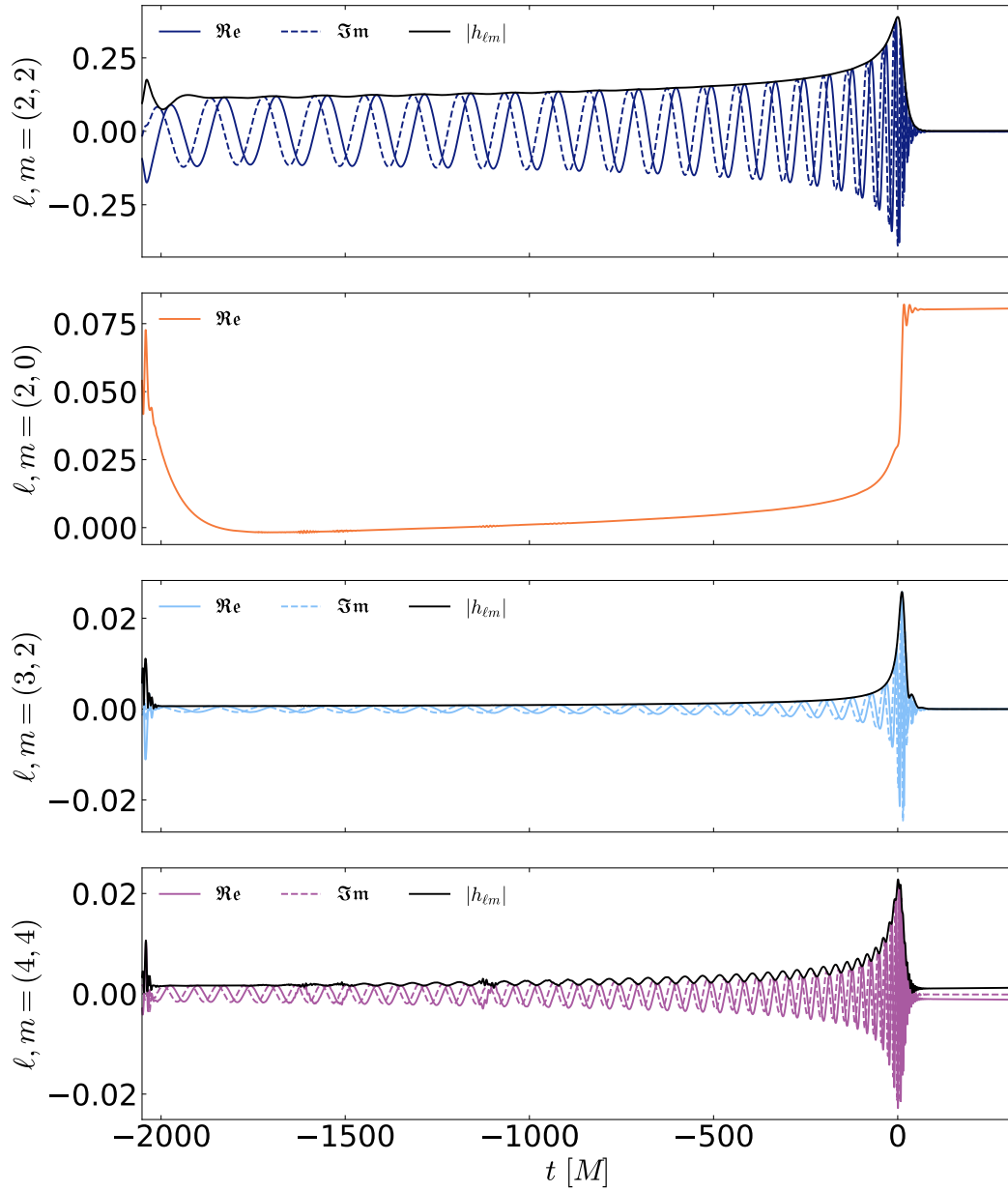


Figure 4.3: GW150914-type GW mapped to the inspiral superrest frame at a time shortly after junk radiation. All waveforms are plotted so $t = 0$ corresponds to this peak of the 2, 2 mode. The four rows are the $\ell, m = (2, 2), (2, 0), (2, 1), (3, 2)$ modes of the waveforms, respectively. We plot the real part, imaginary part, and amplitude for each mode except the 2, 0 mode which is only real.

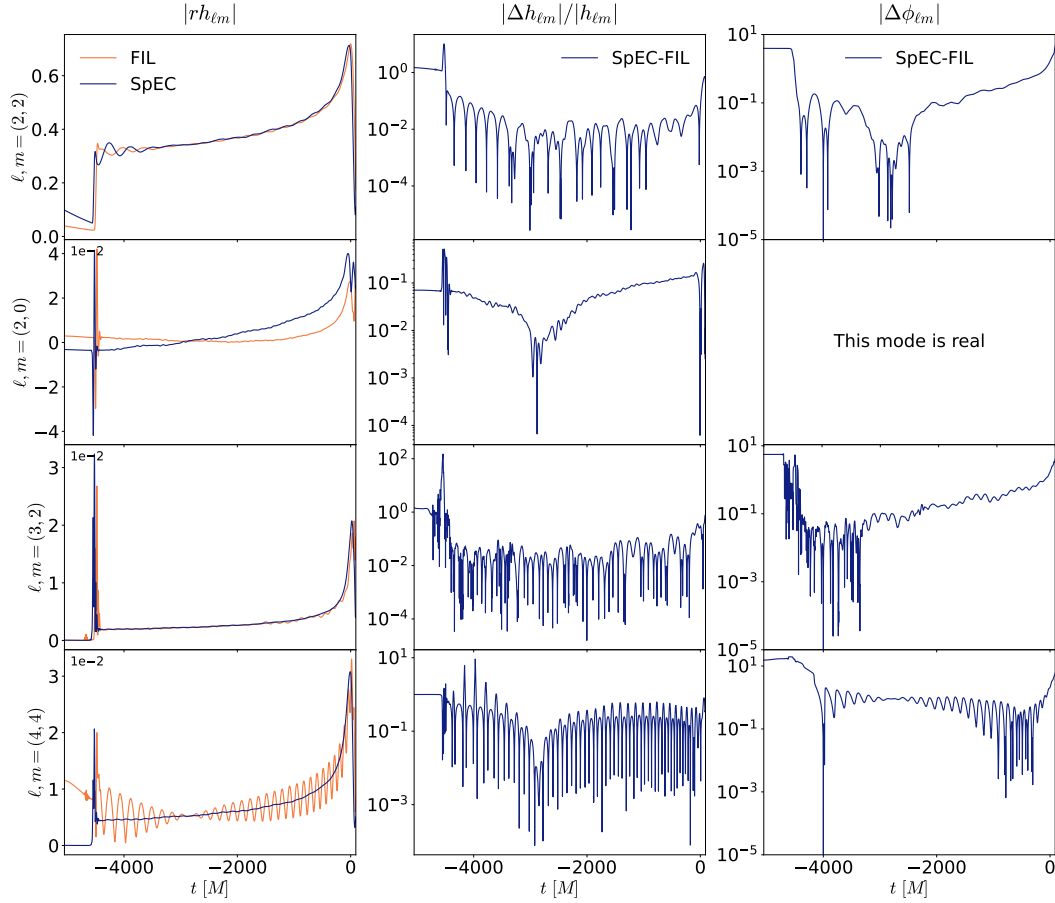


Figure 4.4: Two BNS simulations from FIL and SpEC with an identical starting configuration mapped to SpEC’s inspiral superrest frame at time $t - t_{22}^{\text{peak}} = -3000M$ before the peak of the amplitude of the 2, 2 mode of the waveform. All waveforms are plotted so $t = 0$ corresponds to this peak of the 2, 2 mode. The four rows are the $\ell, m = (2, 2), (2, 0), (3, 2), (4, 4)$ modes of the waveforms, respectively. The left column is the amplitude of each mode. The middle column is the fractional amplitude difference between the two codes. The third column is the phase difference of the waveforms where we zero out the phase of each waveform at $t - t_{22}^{\text{peak}} = -4000M$. Since the $\ell, m = (2, 0)$ isn’t oscillatory (i.e. real), we omit the phase difference plot for this mode.

in Fig. 4.4 were performed with SpEC[46–49] and the Frankfurt-Illinois GRMHD (FIL) [50, 51] codes. The setup for the simulations is equal-mass non-spinning $1.35M_{\odot}$ NSs. Here, we don’t provide an analysis of the waveforms as this work is still in progress, but we simply show this as proof that we can run SpECTRE CCE on non-SXS codes.

4.4.5 Einstein Toolkit

Finally, we show in-progress work of SpECTRE CCE running on worldtube data produced by the Einstein Toolkit [52, 53]. The data was output with the CCE_Export thorn [54], developed specifically to output the worldtube in metric modal format, specifying the gauge information in the CCE input file to compute the time derivatives. In Fig. 4.5, we show some example modes from an equal-mass quasicircular non-spinning BBH simulation which started with separation $12M$. The $(2, 0)$ mode shows a similar post-merger behavior to Fig. 4.1, suggesting more resolution is needed during and after ringdown. Also the amplitude of the $(2, 2)$ mode appears to be oscillating throughout the inspiral. This could potentially be due to some non-zero eccentricity or some mode-mixing. Further study would be needed to understand the glitch in the $(2, 0)$ and $(3, 2)$ mode around time $-1900M$, since this does not coincide with the worldtube radius of $R = 200M$ used here.

4.5 Conclusion

We present additions and improvements to the SpECTRE CCE module that allow for running CCE on the output of other NR codes besides the SXS Collaboration’s SpEC and SpECTRE. SpECTRE CCE itself now only accepts worldtube data in the form of the Bondi scalars from Eqs. 4.25-4.30 represented as complex-valued spherical harmonic coefficients/modes. However, now SpECTRE also provides a conversion executable to put worldtube data into this expected format. This conversion executable accepts the spatial metric, lapse, and shift, along with their spatial derivatives as inputs, plus parameters for choosing which gauge condition the Cauchy evolution was performed with so the conversion executable can compute the time derivatives itself.

These improvements were made to better facilitate comparisons of numerical waveforms between NR codes. First, CCE better captures the non-linear effects of GR compared to extrapolation because the spacetime metric is actually evolved out to $\mathcal{I}^+$. Second, CCE accounts for the different evolution gauges used by NR codes, by computing waveforms in the same frame at $\mathcal{I}^+$ (up to BMS transformations which

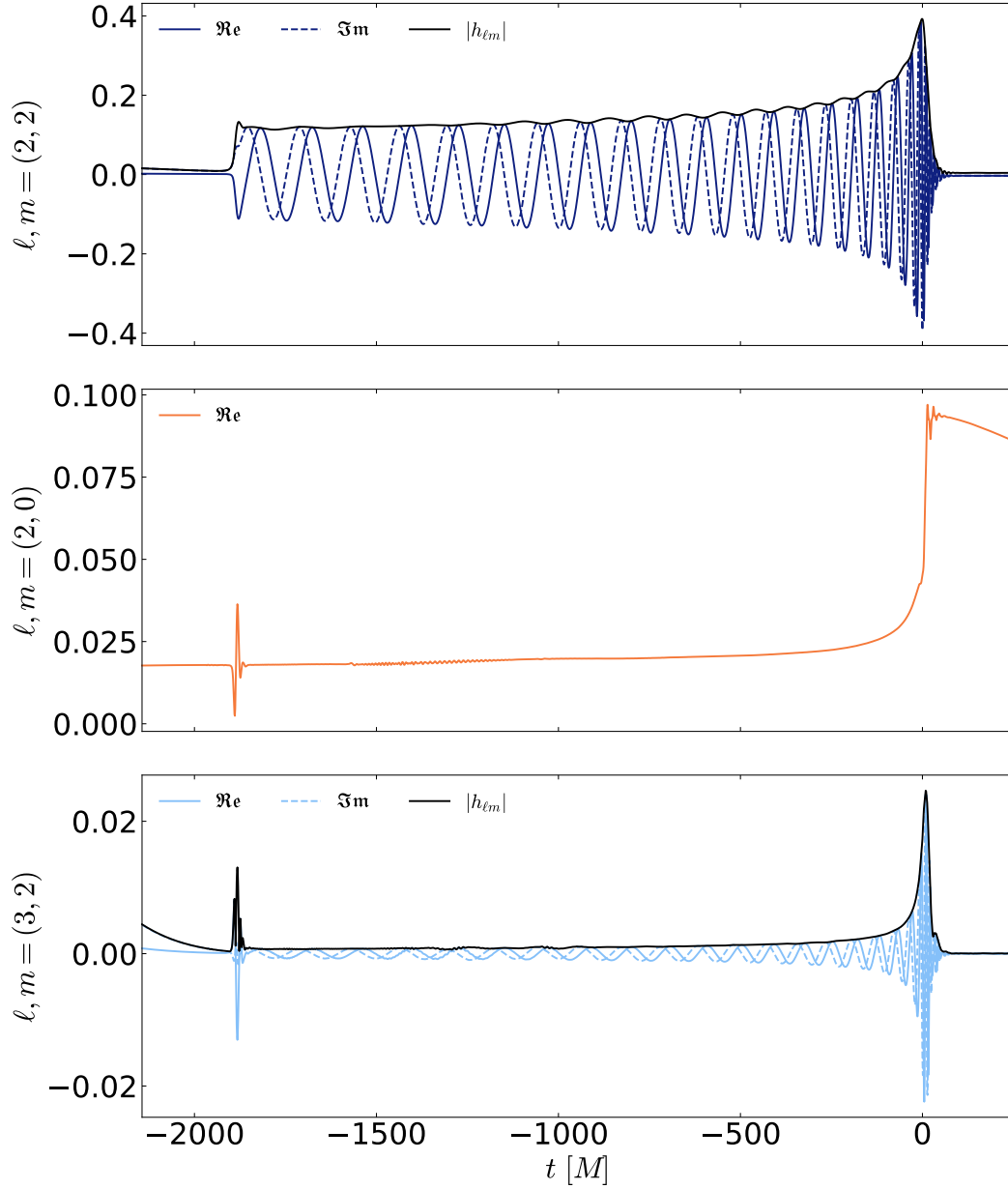


Figure 4.5: Equal-mass non-spinning BBH waveform from the Einstein Toolkit mapped to the inspiral superrest frame at time $t - t_{22}^{\text{peak}} = -1500M$ before the peak of the amplitude of the 2, 2 mode. All waveforms are plotted so $t = 0$ corresponds to this peak of the 2, 2 mode. The four rows are the $\ell, m = (2, 2), (2, 0), (3, 2)$ modes of the waveform, respectively. We plot the real part, imaginary part, and amplitude for each mode except the 2, 0 mode which is only real.

can be freely chosen). Prior to this work, comparing NR waveforms consisted of comparing extrapolated waveforms, assuming they are in the same gauge.

In § 4.4, we present several examples of codes using SpECTRE CCE to extract GWs, along with preliminary frame-fixing analysis of some waveforms. We show CCE running with worldtube data from SpECTRE (§ 4.4.1), SpEC (§ 4.4.2), AthenaK (§ 4.4.3), FIL (§ 4.4.4), and ETK (§ 4.4.5). We give examples for frame-fixing analysis for the SpECTRE and SpEC simulations.

There are a couple avenues for future work available using SpECTRE CCE. First, with SpECTRE specifically, is improving the Cauchy evolution to remove the undesirable features in the waveforms modes seen in Fig. 4.1. This is already underway at Caltech, Cornell, and Cal State Fullerton and will hopefully be ready soon so SpECTRE can produce catalog-ready waveforms.

A second avenue for future work which will benefit all that use SpECTRE CCE is improving the initial data on the first null slice of the characteristic evolution. The imperfect initial data can be seen in the early-time amplitude of the 2, 2 mode in both Fig. 4.1 and 4.2. Significant work has already been put into solving this problem [55], but further investigation is still needed.

A third avenue would be performing a highly detailed study of CCE waveforms from BBH simulations by different NR codes. To do this study, the exact same initial data should be used by each NR code, and interpolated onto the domains. This ensures that the physical parameters for the BBH, e.g. mass ratio and spins, are as close as possible. Gauge dependent parameters such as coordinate separation, angular velocity, etc may be different though. Then, each NR code evolves the BBH through merger to a sufficient time after the BH has rung down. Ideally the resolution of each simulation from the different codes should be roughly the same based on some gauge-independent measure. Then all codes use SpECTRE CCE to extract the GWs at $\mathcal{I}^+$. From here, a number of different frame-fixing techniques can be used to analyze the data such as mapping to the remnant superrest frame, mapping each code's waveform to their own inspiral superrest frame, or mapping each code's waveform to one specific code's inspiral superrest frame. Once the frame fixing is complete, a full analysis of the memory can be done.

Overall, we believe these improvements to the SpECTRE CCE module will benefit the NR community by facilitating more comparison studies amongst the different NR codes and groups. This will drive the accuracy of NR waveforms closer to the

necessary requirements for future GW detectors.

Bibliography

- [1] Geraint Pratten, Sascha Husa, Cecilio Garcia-Quiros, Marta Colleoni, Antoni Ramos-Buades, Hector Estelles, and Rafel Jaume. Setting the cornerstone for a family of models for gravitational waves from compact binaries: The dominant harmonic for nonprecessing quasicircular black holes. *Phys. Rev. D*, 102(6):064001, 2020. doi: 10.1103/PhysRevD.102.064001.
- [2] Cecilio García-Quiros, Marta Colleoni, Sascha Husa, Héctor Estellés, Geraint Pratten, Antoni Ramos-Buades, Maite Mateu-Lucena, and Rafel Jaume. Multimode frequency-domain model for the gravitational wave signal from nonprecessing black-hole binaries. *Phys. Rev. D*, 102(6):064002, 2020. doi: 10.1103/PhysRevD.102.064002.
- [3] Geraint Pratten et al. Computationally efficient models for the dominant and subdominant harmonic modes of precessing binary black holes. *Phys. Rev. D*, 103(10):104056, 2021. doi: 10.1103/PhysRevD.103.104056.
- [4] Mohammed Khalil, Alessandra Buonanno, Hector Estelles, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, and Antoni Ramos-Buades. Theoretical groundwork supporting the precessing-spin two-body dynamics of the effective-one-body waveform models SEOBNRv5. *Phys. Rev. D*, 108(12):124036, 2023. doi: 10.1103/PhysRevD.108.124036.
- [5] Antoni Ramos-Buades, Alessandra Buonanno, Héctor Estellés, Mohammed Khalil, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, and Mahlet Shiferaw. Next generation of accurate and efficient multipolar precessing-spin effective-one-body waveforms for binary black holes. *Phys. Rev. D*, 108(12):124037, 2023. doi: 10.1103/PhysRevD.108.124037.
- [6] Maarten van de Meent, Alessandra Buonanno, Deyan P. Mihaylov, Serguei Ossokine, Lorenzo Pompili, Niels Warburton, Adam Pound, Barry Wardell, Leanne Durkan, and Jeremy Miller. Enhancing the SEOBNRv5 effective-one-body waveform model with second-order gravitational self-force fluxes. *Phys. Rev. D*, 108(12):124038, 2023. doi: 10.1103/PhysRevD.108.124038.
- [7] Mark A. Scheel et al. The SXS collaboration’s third catalog of binary black hole simulations. *Class. Quant. Grav.*, 42(19):195017, 2025. doi: 10.1088/1361-6382/adfd34.
- [8] John David Jackson. *Classical Electrodynamics*. Wiley, 1998. ISBN 978-0-471-30932-1.
- [9] David J. Griffiths. *Introduction to Electrodynamics*. Cambridge University Press, 4th edition, 2017. ISBN 978-1-108-42041-9.

- [10] Charles W. Misner, Kip S. Thorne, and John A. Wheeler. *Gravitation*. W. H. Freeman, San Francisco, 1973. ISBN 978-0-7167-0344-0, 978-0-691-17779-3.
- [11] Robert M. Wald. *General Relativity*. Chicago Univ. Pr., Chicago, USA, 1984. doi: 10.7208/chicago/9780226870373.001.0001.
- [12] Sean M. Carroll. *Spacetime and Geometry: An Introduction to General Relativity*. Cambridge University Press, July 2019. ISBN 978-0-8053-8732-2, 978-1-108-48839-6, 978-1-108-77555-7. doi: 10.1017/9781108770385.
- [13] Robert Abbott et al. GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run. *Phys. Rev. X*, 13(4):041039, 2023. doi: 10.1103/PhysRevX.13.041039.
- [14] Robert Abbott et al. GWTC-2.1: Deep extended catalog of compact binary coalescences observed by LIGO and Virgo during the first half of the third observing run. *Phys. Rev. D*, 109(2):022001, 2024. doi: 10.1103/PhysRevD.109.022001.
- [15] Robert Abbott et al. GWTC-2: Compact binary coalescences observed by LIGO and Virgo during the first half of the third observing run. *Phys. Rev. X*, 11:021053, 2021. doi: 10.1103/PhysRevX.11.021053.
- [16] Benjamin P. Abbott et al. GWTC-1: A gravitational-wave transient catalog of compact binary mergers observed by LIGO and Virgo during the first and second observing runs. *Phys. Rev. X*, 9(3):031040, 2019. doi: 10.1103/PhysRevX.9.031040.
- [17] David Garfinkle. Harmonic coordinate method for simulating generic singularities. *Phys. Rev. D*, 65:044029, 2002. doi: 10.1103/PhysRevD.65.044029.
- [18] Lee Lindblom, Mark A. Scheel, Lawrence E. Kidder, Robert Owen, and Oliver Rinne. A new generalized harmonic evolution system. *Class. Quant. Grav.*, 23:S447–S462, 2006. doi: 10.1088/0264-9381/23/16/S09.
- [19] Takashi Nakamura, Ken-ichi Oohara, and Yasufumi Kojima. General relativistic collapse to black holes and gravitational waves from black holes. *Prog. Theor. Phys. Suppl.*, 90:1–218, 1987. doi: 10.1143/PTPS.90.1.
- [20] Masaru Shibata and Takashi Nakamura. Evolution of three-dimensional gravitational waves: Harmonic slicing case. *Phys. Rev. D*, 52:5428–5444, 1995. doi: 10.1103/PhysRevD.52.5428.
- [21] Thomas W. Baumgarte and Stuart L. Shapiro. On the numerical integration of Einstein’s field equations. *Phys. Rev. D*, 59:024007, 1998. doi: 10.1103/PhysRevD.59.024007.

- [22] Carles Bona, Tomáš Ledvinka, Carlos Palenzuela, and Michal Žáček. General covariant evolution formalism for numerical relativity. *Phys. Rev. D*, 67:104005, 2003. doi: 10.1103/PhysRevD.67.104005.
- [23] Sebastiano Bernuzzi and David Hilditch. Constraint violation in free evolution schemes: Comparing BSSNOK with a conformal decomposition of Z4. *Phys. Rev. D*, 81:084003, 2010. doi: 10.1103/PhysRevD.81.084003.
- [24] David Hilditch, Sebastiano Bernuzzi, Marcus Thierfelder, Zhoujian Cao, Wolfgang Tichy, and Bernd Bruegmann. Compact binary evolutions with the Z4c formulation. *Phys. Rev. D*, 88:084057, 2013. doi: 10.1103/PhysRevD.88.084057.
- [25] Daniela Alic, Carles Bona-Casas, Carles Bona, Luciano Rezzolla, and Carlos Palenzuela. Conformal and covariant formulation of the Z4 system with constraint-violation damping. *Phys. Rev. D*, 85:064040, 2012. doi: 10.1103/PhysRevD.85.064040.
- [26] Bela Szilagyi, Lee Lindblom, and Mark A. Scheel. Simulations of binary black hole mergers using spectral methods. *Phys. Rev. D*, 80:124010, 2009. doi: 10.1103/PhysRevD.80.124010.
- [27] Thomas W. Baumgarte and Stuart L. Shapiro. *Numerical Relativity: Solving Einstein's Equations on the Computer*. Cambridge University Press, 2010. doi: 10.1017/CBO9781139193344.
- [28] Michael Boyle et al. The SXS collaboration catalog of binary black hole simulations. *Class. Quant. Grav.*, 36(19):195006, 2019. doi: 10.1088/1361-6382/ab34e2.
- [29] Oliver Sarbach and Manuel Tiglio. Gauge invariant perturbations of Schwarzschild black holes in horizon-penetrating coordinates. *Phys. Rev. D*, 64:084016, 2001. doi: 10.1103/PhysRevD.64.084016.
- [30] Tullio Regge and John A. Wheeler. Stability of a Schwarzschild singularity. *Phys. Rev.*, 108:1063–1069, 1957. doi: 10.1103/PhysRev.108.1063.
- [31] Frank J. Zerilli. Effective potential for even-parity Regge-Wheeler gravitational perturbation equations. *Phys. Rev. Lett.*, 24:737–738, 1970. doi: 10.1103/PhysRevLett.24.737.
- [32] James Healy and Carlos O. Lousto. Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits. *Phys. Rev. D*, 105(12):124010, 2022. doi: 10.1103/PhysRevD.105.124010.
- [33] Deborah Ferguson et al. Second MAYA catalog of binary black hole numerical relativity waveforms. *Phys. Rev. D*, 112(4):044043, 2025. doi: 10.1103/gk7x-9cds.

- [34] Eleanor Hamilton et al. Catalog of precessing black-hole-binary numerical-relativity simulations. *Phys. Rev. D*, 109(4):044032, 2024. doi: 10.1103/PhysRevD.109.044032.
- [35] Jordan Moxon, Mark A. Scheel, and Saul A. Teukolsky. Improved Cauchy-characteristic evolution system for high-precision numerical relativity waveforms. *Phys. Rev. D*, 102(4):044052, 2020. doi: 10.1103/PhysRevD.102.044052.
- [36] Jordan Moxon, Mark A. Scheel, Saul A. Teukolsky, Nils Deppe, Nils Fischer, Francois Hébert, Lawrence E. Kidder, and William Throwe. SpECTRE Cauchy-characteristic evolution system for rapid, precise waveform extraction. *Phys. Rev. D*, 107(6):064013, 2023. doi: 10.1103/PhysRevD.107.064013.
- [37] Hermann Bondi, M. G. J. Van der Burg, and A. W. K. Metzner. Gravitational waves in general relativity, VII. Waves from axi-symmetric isolated system. *Proc. R. Soc. A*, 269(1336):21–52, August 1962. ISSN 2053-9169. doi: 10.1098/rspa.1962.0161. URL <http://dx.doi.org/10.1098/rspa.1962.0161>.
- [38] Rainer Sachs. Gravitational waves in general relativity. VIII. Waves in asymptotically flat space-time. *Proc. R. Soc. A*, 270(1340):103–126, October 1962. ISSN 2053-9169. doi: 10.1098/rspa.1962.0206. URL <http://dx.doi.org/10.1098/rspa.1962.0206>.
- [39] Keefe Mitman et al. Fixing the BMS frame of numerical relativity waveforms with BMS charges. *Phys. Rev. D*, 106(8):084029, 2022. doi: 10.1103/PhysRevD.106.084029.
- [40] Mike Boyle, Dante Iozzo, and Leo C. Stein. moble/scri: V1.2. 10.5281/zenodo.4041972, September 2020. URL <https://doi.org/10.5281/zenodo.4041972>.
- [41] Nicholas W. Taylor, Michael Boyle, Christian Reisswig, Mark A. Scheel, Tony Chu, Lawrence E. Kidder, and Béla Szilágyi. Comparing gravitational waveform extrapolation to Cauchy-characteristic extraction in binary black hole simulations. *Phys. Rev. D*, 88(12):124010, 2013. doi: 10.1103/PhysRevD.88.124010.
- [42] Nils Deppe, William Throwe, Lawrence E. Kidder, Nils L. Vu, Kyle C. Nelli, Cristóbal Armaza, Marceline S. Bonilla, François Hébert, Yoonsoo Kim, Prayush Kumar, Geoffrey Lovelace, Alexandra Macedo, Jordan Moxon, Eamonn O’Shea, Harald P. Pfeiffer, Mark A. Scheel, Saul A. Teukolsky, Nikolas A. Wittek, Isha Anantpurkar, Carter Anderson, Michael Boyle, Alexander Carpenter, Andrea Ceja, Himanshu Chaudhary, Nicholas Corso, Nora Fayyazuddin Ljungberg, Francois Foucart, Noora Ghadiri, Matthew Giesler, Jason S. Guo, Sarah Habib, Chenhang Huang, Dante A. B. Iozzo, Ken Z. Jones, Guillermo Lara, Isaac Legred, Dongjun Li, Sizheng Ma, Denyz Melchor,

- Iago Mendes, Marlo Morales, Elias R. Most, Michael Murphy, Peter James Nee, Alejandro Osorio, Michael A. Pajkos, Kyle Pannone, Vaishak Prasad, Teresita Ramirez, Noah Ring, Hannes R. Rüter, Jennifer Sanchez, Leo C. Stein, Daniel Tellez, Sierra Thomas, Vittoria Tommasini, Daniel Vieira, Tom Wlodarczyk, David Wu, and Jooheon Yoo. SpECTRE, August 2025. URL <https://doi.org/10.5281/zenodo.16906840>.
- [43] David Radice, Rossella Gamba, Hengrui Zhu, and Alireza Rashti. AthenaK simulations of the binary black hole merger GW150914. *Class. Quant. Grav.*, 42(18):185003, 2025. doi: 10.1088/1361-6382/adfffa.
- [44] Hengrui Zhu, Jacob Fields, Francesco Zappa, David Radice, James M. Stone, Alireza Rashti, William Cook, Sebastiano Bernuzzi, and Boris Daszuta. Performance-portable numerical relativity with AthenaK. *Astrophys. J. Suppl.*, 278(2):50, 2025. doi: 10.3847/1538-4365/adcf96.
- [45] David Radice, Alireza Rashti, and Rossella Gamba. AthenaK tutorial: GW150914, 2025. URL <https://github.com/dradice/athenak-tutorial-gw150914>. Accessed: 2026-04-08.
- [46] Matthew D. Duez, Francois Foucart, Lawrence E. Kidder, Harald P. Pfeiffer, Mark A. Scheel, and Saul A. Teukolsky. Evolving black hole-neutron star binaries in general relativity using pseudospectral and finite difference methods. *Phys.Rev.D*, 78(10):104015, November 2008. doi: 10.1103/PhysRevD.78.104015.
- [47] Francois Foucart, Lawrence E. Kidder, Harald P. Pfeiffer, and Saul A. Teukolsky. Initial data for black hole-neutron star binaries: A flexible, high-accuracy spectral method. *Phys. Rev. D*, 77:124051, 2008. doi: 10.1103/PhysRevD.77.124051.
- [48] Roland Haas et al. Simulations of inspiraling and merging double neutron stars using the spectral Einstein code. *Phys. Rev. D*, 93(12):124062, 2016. doi: 10.1103/PhysRevD.93.124062.
- [49] Francois Foucart et al. Gravitational waveforms from spectral Einstein code simulations: Neutron star-neutron star and low-mass black hole-neutron star binaries. *Phys. Rev. D*, 99(4):044008, 2019. doi: 10.1103/PhysRevD.99.044008.
- [50] Zachariah B. Etienne, Vasileios Paschalidis, Roland Haas, Philipp Mösta, and Stuart L. Shapiro. IllinoisGRMHD: An open-source, user-friendly GRMHD code for dynamical spacetimes. *Class. Quant. Grav.*, 32:175009, 2015. doi: 10.1088/0264-9381/32/17/175009.
- [51] Elias R. Most, L. Jens Papenfort, and Luciano Rezzolla. Beyond second-order convergence in simulations of magnetized binary neutron stars with realistic microphysics. *Mon. Not. Roy. Astron. Soc.*, 490(3):3588–3600, 2019. doi: 10.1093/mnras/stz2809.

- [52] Frank Löffler et al. The Einstein Toolkit: A community computational infrastructure for relativistic astrophysics. *Class. Quant. Grav.*, 29:115001, 2012. doi: 10.1088/0264-9381/29/11/115001.
- [53] Maxwell Rizzo, Roland Haas, Steven R. Brandt, et al. The Einstein Toolkit (the ‘martin d. Kruskal’ release, et_2025_05), 2025. URL <https://zenodo.org/records/15520463>.
- [54] Deborah Ferguson and Noora Ghadiri. CCE_Export, 2025. URL <https://zenodo.org/records/15497984>.
- [55] Guido Da Re et al. Initial data construction for Cauchy-characteristic evolution. In preparation, 2026.