

NUMERICAL METHODS FOR EXASCALE
MAGNETOHYDRODYNAMICS SIMULATIONS IN
GENERAL RELATIVITY

A Dissertation

Presented to the Faculty of the Graduate School
of Cornell University

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

by

William Thomas Throwe

May 2019

© 2019 William Thomas Throwe
ALL RIGHTS RESERVED

TABLE OF CONTENTS

| | |
|--|-----------|
| Table of Contents | 3 |
| 1 Introduction | 7 |
| 1.1 Compact binary inspirals | 7 |
| 1.2 Numerical simulations | 9 |
| 1.3 Current methods | 12 |
| 1.4 Limitations of current methods | 16 |
| 1.5 A next-generation highly parallel code | 18 |
| 1.5.1 Discontinuous Galerkin methods | 19 |
| 1.5.2 Task-based parallelism | 21 |
| 1.6 Local time-stepping | 22 |
| 1.7 GRMHD in SpECTRE | 24 |
| 2 A high-order, conservative integrator with local time-stepping | 26 |
| 2.1 Introduction | 26 |
| 2.2 The method | 30 |
| 2.2.1 Adams-Bashforth methods | 30 |
| 2.2.2 Conserved quantities | 31 |
| 2.2.3 Second-order 2 : 1 stepping | 32 |
| 2.2.4 Conservative time steppers | 34 |
| 2.3 Special cases | 37 |
| 2.3.1 Element splitting | 37 |
| 2.3.2 Two-set case | 38 |
| 2.4 Numerical results | 39 |
| 2.5 Conclusions | 45 |
| 2.A Element splitting for general methods | 48 |
| 2.B Tables of coefficients for 2 : 1 LTS rules | 50 |
| 2.B.1 Order 2 | 52 |
| 2.B.2 Order 3 | 53 |
| 2.B.3 Order 4 | 55 |
| 3 SpECTRE: A Task-based Discontinuous Galerkin Code for Relativistic Astrophysics | 60 |
| 3.1 Introduction | 60 |
| 3.2 Equations | 62 |
| 3.3 Methods | 67 |
| 3.3.1 Discontinuous Galerkin | 67 |
| 3.3.2 Domain | 73 |
| 3.3.3 Limiting | 76 |
| 3.3.4 Time Stepping | 80 |
| 3.3.5 Numerical Fluxes | 80 |
| 3.3.6 Primitive Recovery | 81 |

| | | |
|-------|--|----|
| 3.3.7 | Variable Fixing | 81 |
| 3.3.8 | Parallelism | 83 |
| 3.4 | Results | 84 |
| 3.4.1 | Benchmark tests | 84 |
| 3.4.2 | The Fishbone-Moncrief disk | 94 |
| 3.5 | Acknowledgements | 95 |
| 3.A | Fishbone-Moncrief disk equations | 96 |

NUMERICAL METHODS FOR EXASCALE MAGNETOHYDRODYNAMICS
SIMULATIONS IN GENERAL RELATIVITY

William Thomas Throwe, Ph.D.

Cornell University 2019

This work has two parts. In the first, we treat a problem in evolving partial differential equations numerically. Typical methods for such problems are unstable if the time step is too big. The maximum allowed time step is limited to approximately the information propagation time between spatial grid points. Typical methods also use adaptive mesh refinement: for efficiency, the grid points are more finely spaced only in regions where the solution is rapidly varying. But then these regions have a smaller allowed time step. Using this small time step in regions where it is not needed is itself wasteful. While it would be better to be able to use the small time step only in regions where it is required, developing such local time-stepping methods can be difficult.

We present a family of multistep integrators based on the Adams-Bashforth methods. These schemes can be constructed for arbitrary convergence order with arbitrary step size variation. The step size can differ between different subdomains of the system. It can also change with time within a given subdomain. The methods are linearly conservative, preserving a wide class of analytically constant quantities to numerical roundoff, even when numerical truncation error is significantly higher. These methods are intended for use in solving conservative PDEs in discontinuous Galerkin formulations, but are applicable to any system of ODEs. A numerical test demonstrates these properties and shows that significant speed improvements over the standard Adams-Bashforth schemes can be obtained.

In the second part, we describe a new code, SpECTRE, for solving the GRMHD equations. This code uses the discontinuous Galerkin method and task-based parallelism to achieve scaling to exascale computing clusters. We have demonstrated that the code performs well on a variety of standard GRMHD test problems. We also show partial results from ongoing work evolving a relativistic disk surrounding a black hole.

CHAPTER 1

INTRODUCTION

1.1 Compact binary inspirals

The first direct detection of gravitational waves in September 2015 [1] was a stunning confirmation of general relativity and launched the new era of gravitational wave astronomy. Since then, LIGO and Virgo, a similar detector in Europe, have reported a further ten events. [2] With continuing upgrades to these detectors and the addition of additional detectors around the world, the rate at which events are seen is expected to increase dramatically in the coming years.

All gravitational wave signals observed to date originated from the mergers of astronomical compact objects: black holes and neutron stars. These objects compress enough mass into a small enough region so that Newtonian gravity is no longer a sufficient description of the region around them. When two such objects are in a tight orbit around one another, their gravitational fields interact nonlinearly and the familiar laws of Keplerian orbital motion break down. In this regime, gravitationally bound orbits are no longer stable. A binary system will radiate energy away in the form of gravitational waves and its members will spiral together, initially slowly but eventually in a rapid plunge leading to a collision. These inspiral waves, as well as additional waves released when the objects merge, form the signal detected by our observatories.

The collision of a pair of isolated black holes requires only gravitational physics for its description. Nothing is produced by a binary black hole system that could be observed by more traditional light or particle observatories. If such a system is not completely isolated (for example if the holes have an orbiting disk of matter or an additional stellar companion) it might be possible to observe the gravitational

effects of the black-hole inspiral on the nearby matter or the gravitational lensing of light passing near them, but no such effects have been observed so far.

If at least one of the members of a binary is a neutron star, however, things are very different. A neutron star is made of matter compressed to extreme densities by the star's self gravity. As a result, these objects are characterized by very strong gravitational, nuclear, and electromagnetic forces. When a neutron star is destroyed, either by collision with another star or by being ripped apart by tidal effects, some of its ultra-dense matter is released and can emit electromagnetic and neutrino radiation. Most of this matter falls back in, but some of it is ejected from the system. This matter is extremely neutron rich and forms atomic nuclei with atomic masses much too large for stability. These nuclei will decay to stable products in what is known as the r-process, emitting a characteristic electromagnetic signature. This process is believed to be the primary source of heavy elements in the universe.

When a compact object binary consists of two neutron stars, and in some cases one neutron star and one black hole, the neutron star disruption will produce a wide variety of electromagnetic signals that are bright enough to be observed by traditional telescopes. Such signals were observed to coincide with the end of the gravitational wave signal from one of the LIGO/Virgo events. [3] After the inspiral and merger were observed by the gravitational wave detectors, signals were seen across the electromagnetic spectrum from gamma rays to radio waves. These observations were the first direct evidence that at least some gamma ray bursts are caused by the coalescence of binary neutron-star systems. As more such events are observed, the combination of gravitational and electromagnetic observations will allow measurements ranging from the properties of ultra-dense matter to the rate of expansion of the universe.

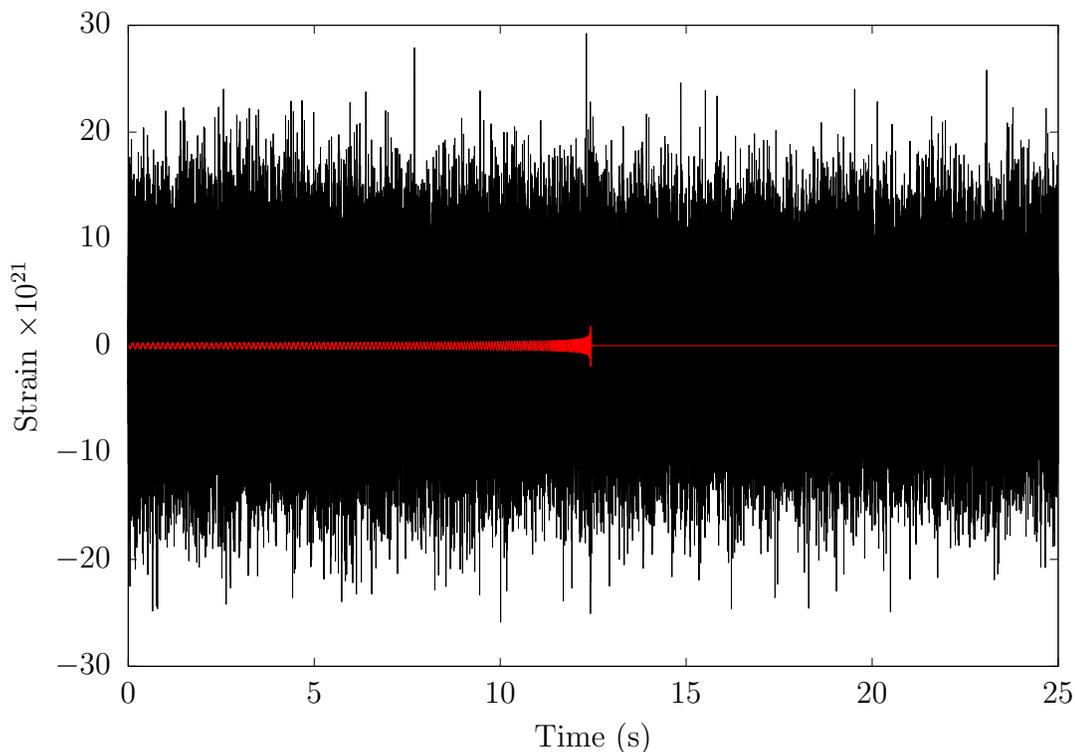


Figure 1.1: A possible gravitational wave signal (red) compared to detector noise (black).

Data source: LIGO Scientific Collaboration

1.2 Numerical simulations

Even for electromagnetic astronomy, there are few cases where a source can be observed in sufficient detail that its signal can be interpreted without at least a basic mathematical model of that source. Light curves (light intensities at different times and frequencies) can be measured, but, particularly for transient events, obtaining more than a basic understanding of the source requires inferring the nature of the matter producing that light, which must generally be done via models. These vary in complexity from simple descriptions that can be evaluated with pen and paper up to full hydrodynamical simulations requiring supercomputers. In any case, parameters in the model can be adjusted to match the observed light as well as possible, and in this way infer the properties of the source.

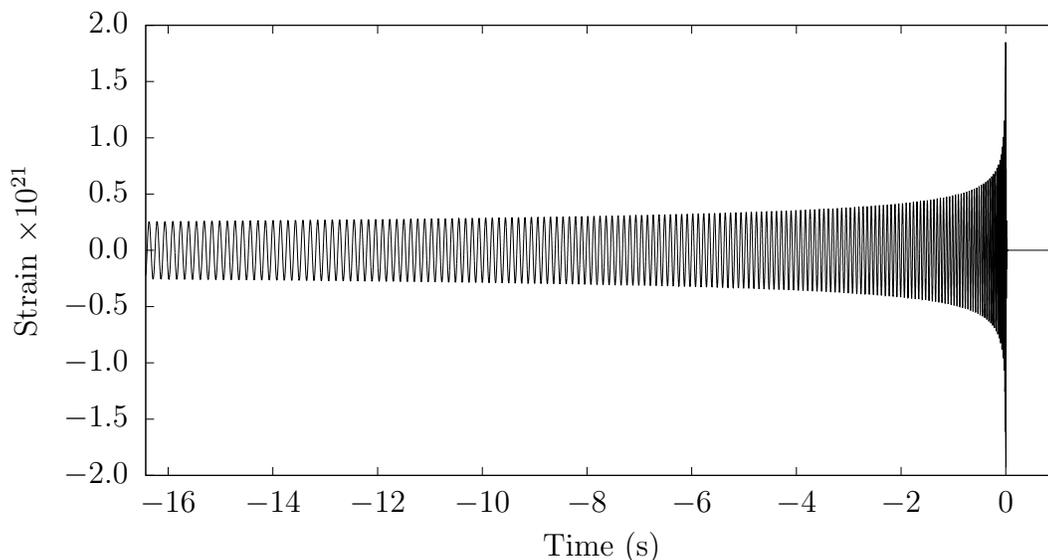


Figure 1.2: A template for a binary merger.

Data source: LIGO Scientific Collaboration

Models are even more necessary for gravitational wave astronomy. In the current generation of detectors, the amplitude of the random detector noise is larger than the amplitude of the typical signal that we hope to detect. (See Figure 1.1 for an example.) The strongest signals can be seen directly in the data, but the noise still makes any direct measurement of a gravitational “light curve” infeasible. Instead, when we identify an event, we generate many likely signals, known as templates, and then see which one matches the data best. A typical template is shown in in Figure 1.2. The random noise from the detector should not look like any signal we expect to see, so, while it will add uncertainty, that uncertainty will be approximately independent of the model we are testing against and it will not prevent us from finding a reasonable best match.

An additional difficulty is detecting an event in the first place. A simple examination of the measured data may not reveal the existence of a signal under the noise. To solve this problem we perform cross-correlations of our detector data with each of our templates at all times. If a template is a good match for the data

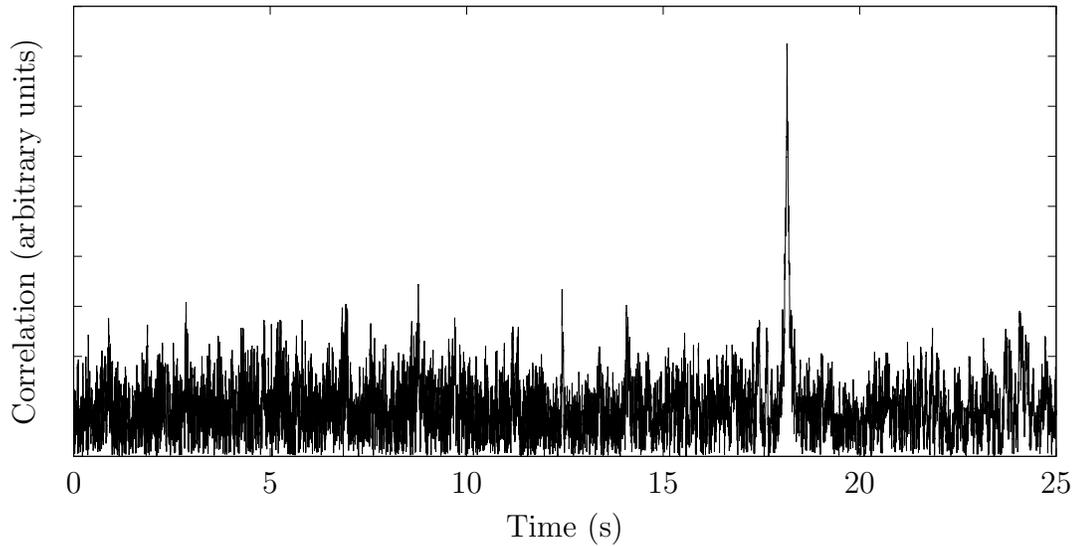


Figure 1.3: Cross-correlation of a template with detector data. The spike indicates a match of the template to the data at a time of around 18 seconds.

Data source: LIGO Scientific Collaboration

we will observe a large spike in the cross-correlation at the time of the event, as shown in Figure 1.3. Once we have identified an event in this way we can determine the event parameters more accurately in a similar manner to electromagnetic events: fine-tuning the model to get the best match. The process of identifying events in a noisy signal by comparing modeled data with measured data for all times is known as matched filtering, and its use is crucial for obtaining useful data from gravitational wave detectors.

This raises the question of what type of model to use. For a compact binary with members that are relatively far apart, the system and its gravitational wave signal can be adequately modeled by adding correction terms to Newtonian gravity. This is known as the post-Newtonian approximation. These equations, while complicated, can be solved by a combination of analytic methods and numerical integration of ordinary differential equations. These post-Newtonian models do an excellent job of describing the long period of slow orbital decay that brings the

objects within a few tens of radii of one another.

Unfortunately, the times where post-Newtonian models are valid for objects heavier than about ten solar masses are generally too far before the binary merger to be visible in detectors like LIGO. More complex semi-analytic models can provide waveforms through times closer to the merger, but all such models lose accuracy as they approach the final orbits and none of them can model the merger itself. To get a full model waveform we must perform numerical evolutions of the Einstein field equations and any matter relevant to the system dynamics. These evolutions are very computationally expensive. There is ongoing research into finding compact descriptions of the known numerical results that can be used to interpolate to cases that have not been simulated, but for a full application to detector data analysis these models still need many more numerical evolutions to work from. (See [4] and references therein.)

1.3 Current methods

The simplest class of methods for numerically integrating partial differential equations (PDEs) is the finite-difference methods. In a finite-difference method, the PDE in question is approximated as a difference equation, that is, derivatives are replaced by approximations obtained by subtracting function values at nearby points. If the function is smooth¹ enough, then as the density of these points increases, the derivative approximations converge to the actual value of the derivative and the error in the numerical solution approaches zero. These methods are relatively robust. The simplest forms make no assumptions about the form of the solution except that there are no features smaller than the point separation.

¹For our purposes, *smooth* can be taken to mean analytic, i.e., at every point a smooth function has a well-defined Taylor series that converges (locally) to the function.

These forms are not very accurate, however. Using the simplest discretization of the derivative using the two adjacent points, the error is proportional to the square of the point separation. Using additional points that are further away can increase the rate of convergence of the solution, but can also cause larger errors when the function is not well approximated by its Taylor expansion across the set of points being used.

Another class of methods is spectral methods. In a spectral method, the solution is represented as an expansion in some collection of basis functions ϕ_k , such as Legendre polynomials or spherical harmonics:

$$f(x) = \sum_{k=0}^{\infty} \hat{f}_k \phi_k(x). \quad (1.1)$$

To obtain a finite numerical representation, this series is truncated after some number of terms

$$f_S(x) = \sum_{k=0}^N \hat{f}_k \phi_k(x). \quad (1.2)$$

For sufficiently smooth functions these series converge exponentially fast as N is increased, so spectral methods can achieve much higher accuracy than finite-difference methods for a similar number of stored values. Furthermore, as the basis functions are known analytically, their derivatives can be calculated exactly and used to obtain high-accuracy derivatives of the numerical solution.

A shortcoming of this approach is that certain common mathematical operations are difficult to perform on a function represented as a spectral expansion. An example of such an operation is the multiplication of two quantities: taking the product of two series is a computationally expensive operation. A common operation requiring this is the integral over the domain of the product of a pair of functions. To overcome this problem, we can introduce a modification to the spectral procedure to produce what is known as a pseudospectral method. The pseudospectral method replaces exact integrals over the domain with Gaussian

quadrature, which is much easier to evaluate. Gaussian quadrature is an approximation to an integral taking the form

$$\int_{\text{domain}} f(x) dx \approx \sum_{i=0}^N w_i f(x_i), \quad (1.3)$$

where the $\{x_i\}$ are a fixed set of points known as the quadrature points. In a pseudospectral method, functions are expanded as a sum of basis functions in the same manner as a true spectral method, but, instead of approximating by truncating this series, an interpolant that matches the function at the quadrature points is used, that is, we define the pseudospectral approximation to be

$$f_{\text{PS}}(x) = \sum_{k=0}^N \tilde{f}_k \phi_k(x) \quad (1.4)$$

with \tilde{f}_k chosen so that

$$f_{\text{PS}}(x_i) = f(x_i) \quad (1.5)$$

at each of the quadrature points. This is less accurate than using a series truncation, but the error converges away exponentially as the number of basis functions is increased, so in practice this is not a concern.

As we are defining the interpolant via its values at the quadrature points, we find that these values are a natural alternative representation of the function. In this new *nodal* representation we store the function values $f_i \equiv f(x_i)$ instead of the expansion coefficients \tilde{f}_k , which are known as the *modal* representation. Using a nodal representation makes performing integrals using (1.3) easy. This representation also allows operations such as multiplication to be carried out pointwise, as in a finite-difference method: $f^2(x)$ is computed as f_i^2 . Converting between nodal and modal representations of a function is a straightforward operation, so we can continue to use advantages of the modal spectral representations, such as easy computation of high-accuracy derivatives using the known analytic forms of the basis functions.

This pair of equivalent representations appears elsewhere in physics. In quantum mechanics, the momentum-space representation of a wavefunction is the modal representation corresponding to the nodal position-space representation using the (infinite dimensional) Fourier basis.

In practice, it is often difficult to find a good collection of basis functions that fits the geometry of a particular problem. For example, for a binary black hole simulation the computational domain consists of a large sphere with two smaller spheres cut out of the inside. To work around this, the computational domain can be divided up into subdomains (also called elements), each of which has a known set of basis functions. The numerical solutions in these elements are then coupled together at the interfaces between them. The convergence of such a method is still exponential, but it is somewhat slower than for a single-element spectral method because the error is sensitive to the number of basis functions in each element, rather than in the whole domain. In practice, the number of functions needed to get a high accuracy solution is usually small enough that this is not a major concern.

A shortcoming of (pseudo)spectral methods is that, while a spectral method converges exponentially for well-behaved functions, not all functions encountered in practice are well behaved. In particular, many equations in physics generate shocks, where the correct solution is discontinuous. In these cases the spectral representation will exhibit Gibbs phenomenon and converge very slowly, or possibly not at all.

The most common method used for numerically solving the Einstein equations is finite differencing. The solutions, however, are known to be smooth, (except at singularities) so the problem should be a good candidate for a spectral approach. The Simulating eXtreme Spacetimes (SXS) collaboration has used these ideas to

develop the Spectral Einstein Code (SpEC), which uses a pseudospectral method to solve the Einstein equations. This has proven to be a good approach for simulating binary black hole inspirals.

Systems containing neutron stars require evolving the equations of general-relativistic hydrodynamics or magnetohydrodynamics (GRMHD). These equations have the unpleasant property that the matter, unlike gravity, can form shocks. At a shock, the solution to the equations is discontinuous in space and cannot be accurately represented as an expansion in smooth basis functions. Equations that can shock are generally solved using finite-difference schemes, which make fewer assumptions about the smoothness of the solution. Even using those methods special care is needed to prevent the numerical solution from “ringing” in a Gibbs-phenomenon-like manner.

SpEC evolves the hydrodynamical equations using a dual-grid method: the fields representing gravity, which are always smooth, are evolved on a pseudospectral grid, while the matter fields are evolved on a finite-difference grid. This allows the gravitational portions of the system to take advantage of the exponential convergence of pseudospectral methods while the matter portions can use the established methods of dealing with shocks in finite-difference codes. The two sets of equations are coupled by interpolating the necessary function values between the two grids.

1.4 Limitations of current methods

To perform full analysis of their data, gravitational wave detectors require the results of many high-accuracy simulations. With current codes, simulating a typical binary black hole merger takes months. Neutron star mergers are slower, but take roughly the same amount of time because they are evolved as much lower resolu-

tion. This is far too slow to enable the full analysis of detector data. For binary black hole systems, while the accuracy of the simulations is typically adequate for current LIGO capabilities, systems with high spins or high mass ratios are still very difficult. For systems containing matter, the accuracy is insufficient even at current detector sensitivity. The situation will only get worse as sensitivities improve.

The accuracy of a simulation can be improved by increasing its resolution. Higher resolution, however, requires more computing power, making the already slow simulations take even longer. The processing power of a single computing core has essentially stagnated since the mid-2000s, so the only way to use more power in a fixed amount of time is through parallelization. Newer supercomputer clusters have more individual computers coupled together, each with more processors, and to use such a cluster efficiently a code must be able to parallelize its work very well.

At a basic level, the problems of parallelizing finite-difference and pseudospectral codes are both the problem of spreading points out across many processors. For a finite-difference method this is, mathematically, straightforward, as the calculation for each step at each point is independent of the calculations (although not the values) at the other points. Spreading these points across a computing cluster does not waste any computation. Unfortunately, communicating the values needed from nearby points is expensive if the points are not on the same processor. For high-order methods, the values from many points are necessary, so the amount of communicated data becomes large if the points are distributed too thinly and slows down the computation.

Pseudospectral methods, on the other hand, achieve higher order by increasing the number of basis functions in each element. Most parts of the calculation only

require the values of other points in the element, so as long as we keep these points together no communication is necessary for those parts. Each element must communicate with each of its neighbors, but this is generally much less communication than for a similar finite-difference method. We have the downside, however, that keeping the data for each element together reduces our options for distributing points, and we may need to divide the domain into more elements to counteract that. The addition of more elements, and to a lesser extent the increase of the order within each element, increases the necessary communication, eventually leading to parallelization efficiency problems.

Current binary black hole simulations in SpEC parallelize well up to a few dozen processors, but after that computation time becomes dominated by evolution of single large elements. For hydrodynamics simulations, SpEC is currently limited by the memory required to store the finite difference grid, which reduces its ability to efficiently use processing power. Hydrodynamics simulations are typically run on thousands of processors.

1.5 A next-generation highly parallel code

Over the next few years, supercomputers will evolve from providing thousands of processors to a typical user to providing millions of processors. As described in the previous section, current codes for astrophysics will not be able to make use of these exascale machines efficiently. Something new is needed. In this section, we describe two key ingredients that our collaboration plans to use in SpECTRE, a new code.

1.5.1 Discontinuous Galerkin methods

To create a code that can efficiently use upwards of thousands of processors, we have chosen an approach based on the discontinuous Galerkin (DG) pseudospectral method. In DG methods the elements are coupled by the exchange of “fluxes” with their neighbors. DG methods are simplest to describe for equations written in conservation form (although they be extended to equations not in conservation form too). For example, a simple equation describing mass conservation is

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v}) \quad \text{or} \quad \frac{d}{dt} \int \rho d^3x = - \oint \rho \mathbf{v} \cdot d\mathbf{S} \quad (1.6)$$

where ρ is the mass density of a fluid and \mathbf{v} is its velocity. The equation describes the rate of change of mass in a volume in terms of the mass flux $\rho \mathbf{v}$ through the boundary. If we take the volume of integration to be a single subdomain, then the fluxes need to be computed at each interface with a neighboring subdomain.

For example, in a hydrodynamics simulation there will be energy and momentum fluxes flowing through each face of each element. In the DG method, the solution in neighboring elements is not required to agree on the boundary between them, (contrasting with the continuous Galerkin method), but the two solutions will be driven together by the requirement that they both use the same value for the flux between them. This makes the coupling between the elements depend only on the nearest neighbors, reducing the amount of communication required when the elements are spread out over multiple processors. This should be contrasted with how high-order finite-difference methods couple together many neighboring elements.

A major advantage of not requiring the solution to be continuous at element boundaries is that when the solution is expanded in basis functions, the basis functions can be local to each element—they do not need to be continuous across the interfaces. This locality also contributes to reducing the communication costs of

the algorithm. Allowing the solution to be discontinuous can be justified mathematically: As long as the error because of the discontinuity is not larger than the error coming from the numerical approximations being made in the interior of each element, and as long as both these errors decrease appropriately as the resolution of the grid is increased, the method is mathematically sound.

In order to spread elements over thousands of processors one needs thousands of elements, and it is preferable that they each have as few neighbors as possible to reduce the required communication. As mentioned previously, in a pseudospectral method smaller elements are less efficient, in that a larger number of total points are needed to achieve a given accuracy. We choose to accept that inefficiency as it allows us to distribute our points to more processors and thereby perform a full evolution in less time, even if the total time used by all processors is increased. As we wish to use small elements, we also avoid using topologically interesting element shapes like spherical shells and instead divide the computational domain into distorted cubes. This allows us to avoid the complexities connected to having a variety of basis functions that might have to change when elements are split.

We want to avoid the dual-grid scheme used in SpEC for simulations with matter because it is expensive and has low accuracy, but we still need to handle shocks in the hydrodynamical quantities. Representing shocks using a spectral basis generally works poorly because the approximate solution does not converge well to the discontinuities as the number of basis functions increases. To get around this shortcoming, we can use elements with a small number of basis functions (often just a linear approximation) and obtain accuracy by making the elements very small. This makes the scheme similar to a finite difference method. By adaptively changing the size of the elements and the number of basis functions in the expansion based on the behavior of the solution, we can get the accuracy of

a pseudospectral method in regions where the solution is smooth but transition to a method that can handle shocks where it is not. The DG algorithm borrows techniques from finite-difference schemes for computing fluxes between element boundaries that can handle shocks.

1.5.2 Task-based parallelism

Orchestrating a simulation spread across thousands of processors is not an easy task. If elements are distributed improperly, the entire simulation can end up waiting for a single processor that has too much work assigned to it. The amount of work required to evolve an element can be difficult to predict ahead of time and elements may be created and destroyed to adjust the accuracy in different parts of the domain, both of which complicate finding a good distribution of elements.

The second key innovation in our new code is to solve this problem using task-based parallelism. In this paradigm, the full computation is divided up into small tasks. For a DG system, examples of tasks could be to advance the solution in one element one step in time, to compute the flux between a pair of elements, or to make a measurement such as the value of the magnetic pressure and output it to disk. Tasks depend on one another: a measurement cannot be made until the quantity to be measured has been computed and a time step cannot be taken until fluxes have been computed. At any given time, there are (ideally many) tasks that are ready to be performed because all the tasks they depend on are complete. In a maximally parallelized scheme, these tasks would simply be distributed to processors as soon as those processors became idle. Going to this extreme is a poor choice for DG simulations, because each task requires a non-negligible amount of data and it is inefficient to transfer that data to whatever processor the task gets assigned to. We therefore assign each task to the processor that has the required data. This

opens the scheme up again to inefficiencies due to an imbalance in the workload across processors, but it also provides a method for fixing such imbalances. If one processor has a backlog of tasks and is slowing the simulation down, we can move some of those tasks, including the data they need, to another processor. This provides a method for a basic form of load balancing that does not need careful tuning to the specific problem at hand.

In theory, this task-based paradigm can make the simulation globally asynchronous, i.e., the current state of the computation may be at very different simulation times at different parts of the computational domain as long as those parts are far enough apart that those times are causally disconnected. In practice, most complex problems will contain global synchronization points: tasks that require the entire simulation to be at the same time. Such tasks might include, for example, changes to the layout of the DG elements. When one of these tasks is necessary, all processors will be forced to wait for the slowest part of the simulation, resulting in wasted computational resources. The frequency of these events must be kept small for a highly parallel code to be efficient.

1.6 Local time-stepping

In many simulations, the typical length scale for interesting phenomena can vary greatly throughout the computational domain. In a GRMHD inspiral, for example, the length scale for gravitational effects is set by the spacetime curvature, which can vary by orders of magnitude between the compact objects and the outer edge of the domain, and the length scale for fluid effects can be very short where there are shocks and other complex fluid flows. To accurately simulate these phenomena, we need to space our grid points at separations smaller than the structures we wish to resolve. For efficient computation, however, we do not want the point spacing

to be too much smaller than the structure, because then we would be wasting resources finely sampling a smooth function. Because of the variation in length scales, these requirements cannot generally be met using a uniform point spacing throughout the domain.

We use a variety of methods to vary the point spacing. The most flexible of these is known as adaptive mesh refinement (AMR). In this scheme, if we need more resolution within some element, we can refine it in one of two ways: we can add more points to the element or we can split the element into multiple smaller elements with a larger total number of points. By repeatedly splitting elements we can achieve resolutions much higher than the overall resolution of the simulation in some regions, allowing an efficient placement of points. This point distribution can be reevaluated and adjusted as the simulation progresses to keep the points well-distributed.

AMR provides a method for efficiently choosing the locations in space of evaluation points, but it does not address the choice of their locations in time. Just as for mesh spacing, there are competing interests when choosing evaluation frequency. For maximum computational efficiency, we would like to space our evaluation times out as much as possible, but choosing too large a spacing can lead to a (sometimes catastrophic) loss of accuracy.

Most commonly, the maximum step size is set not by the timescale of the system being evolved, but by the stability of the integration scheme. The allowed step size is described by the Courant-Friedrichs-Lewy (CFL) condition. For a first-order finite-difference scheme, the CFL condition requires that information (defined by the PDE being solved) not propagate more than one grid spacing per time step. This is a causality requirement: in such a scheme the value at a point depends only on its nearest neighbors at the previous step, so any information

moving faster than one grid point per time step could not be reflected in the numerical solution. The numerical details of the CFL condition are different in pseudospectral methods, but the result that the time step must be restricted to a value approximately proportional to the grid spacing remains.

When applied to a code using AMR, the CFL condition represents a significant efficiency problem. In order for the scheme to be stable, the CFL condition must be satisfied everywhere in the domain, so the time step will be set by the most refined region. This can result in many more steps being taken in less-refined areas than would be required for stability there. The extra accuracy obtained from the smaller time steps is not useful because the time-stepping error in these regions is not usually the dominant source of error.

The solution to this problem is to take the ideas from AMR, but apply them to the temporal locations of the evaluations instead of the spatial ones. If we can choose time steps locally, then we can choose steps comparable to the CFL limit everywhere, avoiding the extra evaluations occurring when the step size is chosen globally. Such a scheme is known as a local time-stepping (LTS) scheme. An LTS integrator must provide a prescription for advancing the solution at a point to a new time step when values at neighboring points are not available. Doing this in a way that does not result in a significant decrease in accuracy compared to a similar global time-stepping method is difficult. We have developed a new method for doing this that has several attractive properties. The method is presented in Chapter 2.

1.7 GRMHD in SpECTRE

A generic feature of the regions surrounding massive bodies is a hydrodynamical disk. The most extreme environments expected to house disks are the regions sur-

rounding black holes and neutron stars. In these environments, disks are expected to result from processes such as neutron-star mergers, accretion from companion stars, or the tidal disruption of nearby objects. These relativistic disks generate extremely strong magnetic fields, and so must be simulated using the GRMHD equations.

The details of the important processes governing the evolution of a disk around a compact object are not well understood. The physical processes underlying the dynamics of such a disk are believed to operate on a vast range of length scales, from the size of the disk itself down to scales dominated by turbulent flow. The disk matter is expected to form a complex set of interacting shocks. In some cases, the magnetic fields coupled to the disk can drive a relativistic jet, although the required conditions are not well understood. It is believed that the time scales necessary to observe large-scale phenomena such as jets will be much longer than the time scales of the smaller flow interactions.

Simulation of disks and other extreme GRMHD systems cannot be done sufficiently accurately by current codes. The SpECTRE project is applying the techniques described in Section 1.5 to develop a new fast, highly parallelizable code for solving the GRMHD equations. SpECTRE can currently simulate a portion of a disk; some technical issues that we believe we understand remain to be dealt with before a full disk can be treated. We expect that as exascale machines become available over the next few years, SpECTRE will be able to treat relativistic disks at the resolution necessary to resolve both the important small-scale physics and the orbital dynamics simultaneously. A description of SpECTRE, as well as the results of its application to the disk and several standard GRMHD test problems, is found in Chapter 3.

CHAPTER 2

A HIGH-ORDER, CONSERVATIVE INTEGRATOR WITH LOCAL TIME-STEPPING¹

2.1 Introduction

A common problem in computational fields is to find approximate solutions to partial differential equations (PDEs). For hyperbolic PDEs, where a solution typically describes an evolution of one or more fields through time, the most common approach is to apply the method of lines, where the spatial coordinates in the PDE are discretized, producing a large system of coupled ordinary differential equations (ODEs). These systems of equations can then be discretized in time and solved using standard explicit integration schemes.

To obtain correct solutions to these equations, the time discretization must be fine enough for the integration to be stable. For a method-of-lines system, this limit is primarily because of the Courant-Friedrichs-Lewy (CFL) condition, which limits the step size to approximately the information propagation time between grid points. The resulting step size can show large variation across the spatial domain because of changes in the propagation speed or, more commonly, because of changes in the spacing of the evaluation points. It is often desirable to increase the spatial resolution in some regions to resolve rapidly varying parts of the solution, but this then restricts the step size allowed for stability. Furthermore, in order to evaluate the system right-hand side, it is necessary to know the entire state of the system at the time of interest. The time step for the whole system is then set by the most restrictive of the conditions over the entire domain. If the problematic points make up a small fraction of the system, then the forced evaluations at the

¹This chapter is based on a manuscript that will be submitted for publication shortly.

remaining points can dominate the computational expense.

To reduce the computational cost of finding these solutions, we would like to evaluate each point at intervals set by its own stability limit, rather than the smallest limit for all the points. A method allowing this is known as a *local time-stepping* (LTS) (or *multirate*) method, as opposed to a global time-stepping (GTS) method. Such a method must describe an update scheme for the frequently evaluated degrees of freedom that does not require knowing the full state of the system.

Modifying a GTS method into an LTS one can have significant drawbacks. The individual steps near locations of time step changes are typically more expensive than for a GTS method, so the benefit of fewer derivative evaluations must outweigh this overhead. Care must be taken when calculating the CFL limit near step size changes to take into account variations in the characteristic speeds of the system in the neighborhood of the element. [5] Furthermore, modifying the GTS scheme can destroy numerically desirable properties of the integrator, such as a high convergence order. LTS schemes also do not naturally provide exact conservation of linear conserved quantities [6], a property often taken for granted for GTS integrators. In a physical system, errors accumulated in these quantities (which can represent, for example, total mass) can produce an approximate solution qualitatively different from the true solution.

Early LTS schemes (for example [7]) typically used GTS integrators with different time steps and performed interpolation to obtain data at times at which it was not produced directly. Such schemes are easy to adapt to arbitrary mesh configurations and can be constructed to obtain the same convergence order as the underlying GTS method, but they do not preserve conserved quantities of the system. Corrections to more accurately treat conservation laws were developed [8], but still only resulted in approximate conservation.

More recently, many methods have been investigated as starting points for more sophisticated LTS methods, including both substep [9–14] and multistep [6, 15, 16] integrators and also less common methods such as leapfrog [15, 17], Richardson extrapolation [18], ADER [19], and implicit methods [20]. Demirel *et al.* [21] have even explored LTS schemes constructed from multiple unrelated GTS integrators. Recently, Günther and Sandu [13] presented a very general family of multirate Runge-Kutta-like methods based on the GARK family of integrators [22] that unifies many of the previous Runge-Kutta-based LTS schemes. These methods are applicable to any problem and can be constructed to have any order of accuracy, but they are not conservative. Sandu and Constantinescu [6] presented an Adams-Bashforth-based scheme based on evaluating the right-hand side of the evolution equations using a combination of data at different times. This system is conservative and applicable to any system of equations, but the method is limited to second-order accuracy at times at which all degrees of freedom are evaluated and first-order accuracy at intermediate times.

LTS integrators for the special case of linear systems have been developed based on Adams-Bashforth [15], Runge-Kutta [11, 14], and leapfrog [15, 17] schemes. Of particular interest here, starting from the Adams-Bashforth methods, Grote and Mitkova [15] found a family of high-order, conservative methods for integral ratios between step sizes on different degrees of freedom. These methods use the linearity of the system to split the equations into a form resembling multiple copies of the standard Adams-Bashforth method.

Some authors have derived methods specialized to the discontinuous Galerkin or finite volume formalisms. The structure of elements coupled comparatively weakly in a standard way by exchange of fluxes allows for some simplifications to the problem. Winters and Kopriva [16] presented a scheme using dense output of the

integrators for each element to calculate fluxes at intermediate times. This scheme is high-order and allows for arbitrary step ratios and varying time steps, but it sacrifices the conservative nature of its parent scheme. Gassner *et al.* [10] presented a similar method, but restored conservation by treating the element and flux terms as a predictor and corrector. Krivodonova [9] constructed a method based on a Runge-Kutta integrator which, while not naturally conservative, was made so by adding a correction to cancel any error in conservation whenever neighboring cells are aligned in time. Cavalcanti *et al.* [23] considered the addition of nonlinear operations, such as slope limiting, to the integration step.

In this paper we present a generic, high-order, conservative scheme based on the Adams-Bashforth family of explicit multistep methods. The method uses the idea of performing single right-hand side evaluations using values from different times, in a similar manner to previous work presented by Sandu and Constantinescu [6]. The scheme is conservative and has the same convergence order as the Adams-Bashforth integrator it is based on. The method allows for generic ratios of step sizes between different degrees of freedom, as well as for arbitrarily varying the individual degrees of freedom's step sizes in time. While the applications discussed here are to discontinuous Galerkin systems, the method is fully general and can be applied to any set of coupled ODEs. When applied to a linear system with integral step size ratios, this scheme reduces to the Adams-Bashforth-based scheme presented by Grote and Mitkova [15].

The remainder of this paper is structured as follows: Section 2.2 presents a derivation of the integration scheme. Section 2.3 discusses simplifications that are applicable when the method is applied to some common special cases. Section 2.4 applies the method to numerical test cases. An appendix lists specific formulas for methods of order 2, 3, and 4.

2.2 The method

2.2.1 Adams-Bashforth methods

Suppose we wish to numerically solve a set of coupled first-order ordinary differential equations

$$\frac{d\mathbf{y}}{dt} = \mathbf{D}(\mathbf{y}), \quad (2.1)$$

where $\mathbf{D}(\mathbf{y})$, the time-derivative operator, is the right-hand side evaluated when the system is in state \mathbf{y} . A common method is to solve for the variables at a (monotonic) sequence of times t_0, t_1, \dots using a k th-order Adams-Bashforth method

$$\Delta\mathbf{y}_n = \Delta t_n \sum_{j=0}^{k-1} \alpha_{nj} \mathbf{D}(\mathbf{y}_{n-j}) \quad (2.2)$$

with $\Delta\mathbf{y}_n = \mathbf{y}_{n+1} - \mathbf{y}_n$, $\Delta t_n = t_{n+1} - t_n$ and the coefficients corresponding to the step given by [16]

$$\alpha_{nj} = \frac{1}{\Delta t_n} \int_{t_n}^{t_{n+1}} dt \ell_j(t; t_n, t_{n-1}, \dots, t_{n-(k-1)}). \quad (2.3)$$

Here

$$\ell_n(t; t_0, \dots, t_{k-1}) = \prod_{\substack{j=0 \\ j \neq n}}^{k-1} \frac{t - t_j}{t_n - t_j} \quad (2.4)$$

are Lagrange polynomials. We will not concern ourselves here with the process of starting the evolution, that is, evaluating $\Delta\mathbf{y}_n$ for $n < k - 1$.

If different degrees of freedom require different time steps for stability, it may be desirable to evaluate these variables at different frequencies, in order to avoid unnecessary computations for the more stable variables. Suppose we divide \mathbf{y} into S sets $\mathbf{y}^1, \dots, \mathbf{y}^S$, and that we wish to evaluate \mathbf{y}^s at times t_0^s, t_1^s, \dots . We can then split (2.1) into an equation for each of these sets:

$$\frac{d\mathbf{y}^s}{dt} = \mathbf{D}^s(\mathbf{y}^1, \dots, \mathbf{y}^S), \quad (2.5)$$

where \mathbf{D}^s is the result of \mathbf{D} restricted to the set s . Any attempt to use this equation to perform an LTS evolution immediately encounters the problem that evaluating its right-hand side requires knowing the entire state of the system, which conflicts with the goal of independent evaluation times for different degrees of freedom.

2.2.2 Conserved quantities

A linear conserved quantity is a quantity C expressible as an inner product of a vector \mathbf{c} with the evolved variables (treated as a vector)

$$C = \mathbf{c} \cdot \mathbf{y}, \tag{2.6}$$

with

$$\mathbf{c} \cdot \mathbf{D}(\mathbf{y}) = 0 \tag{2.7}$$

for all values of \mathbf{y} . Such a quantity is constant under exact integration of the system and under integration using Euler's method. An integrator is called (linearly) *conservative* if all such quantities remain constant when integrating a system using it [24]. It is desirable for an integrator to keep such quantities precisely constant (up to roundoff error) rather than merely constant up to the truncation error of the scheme. Such quantities often have an intuitive physical meaning, and frequently even a small rate of drift can cause qualitative changes in the evolution of the system.

When solving a PDE representing a physical system, the most common linear conserved quantities are integrals over the computational domain of fields representing densities. The vector \mathbf{c} in these cases is the vector of coefficients necessary to perform a numerical integral. In a discontinuous Galerkin scheme these coefficients would combine quadrature weights on the elements and factors arising from coordinate mappings of the elements relative to their canonical shapes.

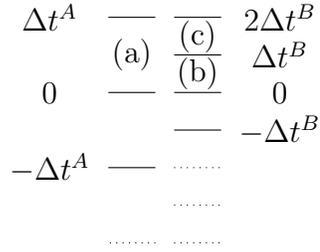


Figure 2.1: The step pattern for a 2 : 1 method on two sets, with time steps $\Delta t^A = 2\Delta t^B$. There are three types of steps: the large step on element A marked (a), and the two types of small step on B marked (b) and (c). For a second order method, we use only the two most recent values of the variables when taking a step. Steps whose values are no longer needed for the indicated steps are marked with dotted lines.

2.2.3 Second-order 2 : 1 stepping

Let us first consider as an example the case of a second-order scheme on two sets, A and B , with B being evaluated twice as often as A . Call their step sizes Δt^A and $\Delta t^B = \Delta t^A/2$. This step pattern is shown in Figure 2.1, where for simplicity we consider the steps starting from $t = 0$ leading up to $t = \Delta t^A$. There are three types of steps to consider: the large step on set A , labeled (a), and the first and second halves of that step on set B , labeled (b) and (c). This case is considered in Sandu and Constantinescu [6], but the method presented there only provides a second-order value when sets have stepped to the same time; intermediate values are only accurate to first order.

We will start with the small step (b). For a GTS Adams-Bashforth method, this step would be given by

$$\Delta \mathbf{y}_b^B = \Delta t^B \left[\frac{3}{2} \tilde{\mathbf{D}}^B(0) - \frac{1}{2} \tilde{\mathbf{D}}^B(-\Delta t^B) \right], \quad (2.8)$$

where we write $\tilde{\mathbf{D}}^B$ instead of \mathbf{D}^B to remind ourselves that we cannot generally obtain these values from a straightforward application of the derivative operator. At time $t = 0$ we do have values over our entire system, so $\tilde{\mathbf{D}}^B(0)$ can, in fact, be evaluated by a simple use of the derivative operator, giving $\tilde{\mathbf{D}}^B(0) = \mathbf{D}^B[\mathbf{y}^A(0), \mathbf{y}^B(0)]$.

We cannot evaluate $\tilde{\mathbf{D}}^B(-\Delta t^B)$ in this manner, because we do not have data for \mathbf{y}^A at $t = -\Delta t^B$, so we must construct it from the values at $t = 0$ and $t = -2\Delta t^B$. There are two reasonable choices of how to do this: average the known values of \mathbf{y}^A to get a value at the desired time and use that to apply the derivative operator, or apply the derivative operator at both times (using the value of \mathbf{y}^B at $-\Delta t^B$ both times) and average the results. We choose the latter, giving step (b) as

$$\Delta \mathbf{y}_b^B = \Delta t^B \left[\frac{3}{2} \mathbf{D}^B[\mathbf{y}^A(0), \mathbf{y}^B(0)] - \frac{1}{4} \mathbf{D}^B[\mathbf{y}^A(0), \mathbf{y}^B(-\Delta t^B)] - \frac{1}{4} \mathbf{D}^B[\mathbf{y}^A(-\Delta t^A), \mathbf{y}^B(-\Delta t^B)] \right]. \quad (2.9)$$

The error in averaging the derivatives is of order $(\Delta t^B)^2$, so it introduces an error of order $(\Delta t^B)^3$ in the value after the step, preserving the second-order quality of the base GTS method.

The second small step, (c), proceeds similarly, except that we now use a derivative at Δt^B instead of $-\Delta t^B$. Instead of averaging the derivatives at different \mathbf{y}^A we must therefore perform a (linear) extrapolation to obtain our approximate derivative $\tilde{\mathbf{D}}^B(\Delta t^B)$. After doing this, we obtain the rule

$$\Delta \mathbf{y}_c^B = \Delta t^B \left[\frac{9}{4} \mathbf{D}^B[\mathbf{y}^A(0), \mathbf{y}^B(\Delta t^B)] - \frac{3}{4} \mathbf{D}^B[\mathbf{y}^A(-\Delta t^A), \mathbf{y}^B(\Delta t^B)] - \frac{1}{2} \mathbf{D}^B[\mathbf{y}^A(0), \mathbf{y}^B(0)] \right]. \quad (2.10)$$

We could use the same procedure to evaluate the large step (a), but, as this would not take into account the value $\mathbf{y}^B(\Delta t^B)$ used for taking the second small step, there is no way this procedure could be conservative. This, however, gives us a hint as to how to proceed: we treat the large step as having two internal steps, one for balancing each of the small steps. In fact, in order to remain conservative, we must take each of these internal steps using the same scheme as for the corresponding small step, except using the part of the derivative corresponding

to set A . This can be seen by considering a generic pair of methods for a step: $\Delta \mathbf{y}^{A,B} = \sum_i k_i^{A,B} \mathbf{D}_i^{A,B}$. The change in a linear conserved quantity during that step is

$$\Delta C^A + \Delta C^B = \mathbf{c}^A \cdot \sum_i k_i^A \mathbf{D}_i^A + \mathbf{c}^B \cdot \sum_i k_i^B \mathbf{D}_i^B = \sum_i [k_i^A \mathbf{c} \cdot \mathbf{D}_i + (k_i^B - k_i^A) \mathbf{c}^B \cdot \mathbf{D}_i^B]. \quad (2.11)$$

The first term vanishes by (2.7), so the only way for two sets to take equal-sized steps in a conservative manner is if they use the same step rule. The procedure for the large step can therefore be found by summing (2.9) and (2.10), giving

$$\begin{aligned} \Delta \mathbf{y}_a^A = & \\ \Delta t^A \left[\frac{9}{8} \mathbf{D}^A[\mathbf{y}^A(0), \mathbf{y}^B(\Delta t^B)] + \frac{1}{2} \mathbf{D}^A[\mathbf{y}^A(0), \mathbf{y}^B(0)] - \frac{1}{8} \mathbf{D}^A[\mathbf{y}^A(0), \mathbf{y}^B(-\Delta t^B)] \right. & \\ \left. - \frac{3}{8} \mathbf{D}^A[\mathbf{y}^A(-\Delta t^A), \mathbf{y}^B(\Delta t^B)] - \frac{1}{8} \mathbf{D}^A[\mathbf{y}^A(-\Delta t^A), \mathbf{y}^B(-\Delta t^B)] \right]. & \quad (2.12) \end{aligned}$$

Note that the coefficients have changed by a factor of 2 compared to the previous equations because of the change of the leading coefficient to Δt^A . As the two small steps were accurate to second order and this is effectively their concatenation, it is also accurate to second order.

2.2.4 Conservative time steppers

Let us return now to the task of finding a general conservative, high-order LTS integrator. First, we will consider the implications of requiring an Adams-Bashforth-like LTS scheme to be conservative. For such a scheme it only makes sense to evaluate (2.6) at times at which all the degrees of freedom are evaluated. We therefore introduce a new quantity $\tilde{\mathbf{y}}_n$ that is defined for the entire set of degrees of freedom for each time \tilde{t}_n at which any set is evaluated, and is equal to \mathbf{y} where the latter is defined. If we provide an update rule for $\tilde{\mathbf{y}}_n$ then, as long as portions

of $\tilde{\mathbf{y}}_n$ that we do not wish to evaluate are never used, we can obtain an LTS method by summing the changes in $\tilde{\mathbf{y}}$ on each of the sets between evaluations. Furthermore, if the step from $\tilde{\mathbf{y}}_n$ to $\tilde{\mathbf{y}}_{n+1}$ is conservative, then the implied full method will be as well.

The condition for this small step to be conservative is

$$0 = \mathbf{c} \cdot \Delta\tilde{\mathbf{y}}_n. \quad (2.13)$$

This is satisfied if we evaluate $\Delta\tilde{\mathbf{y}}_n$ using a standard Adams-Bashforth method, but that would require values of $\tilde{\mathbf{y}}$ that are not included in \mathbf{y} . Comparing (2.7) and (2.13), we see that we will obtain a conservative method if we take

$$\Delta\tilde{\mathbf{y}}_n = \sum_i \beta_{ni} \mathbf{D}(\mathbf{y}_{ni}^1, \dots, \mathbf{y}_{ni}^S) \quad (2.14)$$

for some set of coefficients β_{ni} and with $\mathbf{y}_{ni}^s = \mathbf{y}^s(t)$ for some time t at which set s is evaluated. The choices of these coefficients are not unique, but there is a natural choice. We evaluate each step using a standard order- k Adams-Bashforth scheme, but instead of using the derivatives of the function that we cannot evaluate, we use approximate derivatives $\tilde{\mathbf{D}}_n$. As long as these are accurate to order $k - 1$, we will lose no formal accuracy for the step. We evaluate $\tilde{\mathbf{D}}_n$ by treating $\mathbf{D}(\mathbf{y}^1(t_1), \dots, \mathbf{y}^S(t_S))$ as a function of the times t_1, \dots, t_S independently, and then performing a multidimensional interpolation from known values. To obtain the required accuracy, we need evaluations at at least k times from each set, and it is natural to choose the most recent values. The known values of \mathbf{D} then form a lattice in the multidimensional space. Multidimensional interpolation from such a lattice is not unique, but a natural choice is to perform it as a series of

one-dimensional interpolations.² Combining all these ideas, we have

$$\Delta \tilde{\mathbf{y}}_n = \sum_{q^1=0}^{k-1} \cdots \sum_{q^S=0}^{k-1} I_{n;q^1 \dots q^S} \mathbf{D}(\mathbf{y}_{m_n^1 - q^1}^1, \dots, \mathbf{y}_{m_n^S - q^S}^S), \quad (2.15)$$

where m_n^s is defined by $t_{m_n^s}^s \leq \tilde{t}_n < t_{m_n^s+1}^s$, i.e., it is the index of the last evaluation on set s that can influence $\Delta \tilde{t}_n$ (see Figure 2.2). The coefficients of the derivatives are

$$I_{n;q^1 \dots q^S} = \Delta \tilde{t}_n \sum_{i=0}^{k-1} \tilde{\alpha}_{ni} \prod_{s=1}^S \ell_{q^s}(\tilde{t}_{n-i}; t_{m_n^s}^s, \dots, t_{m_n^s - (k-1)}^s), \quad (2.16)$$

with $\tilde{\alpha}_{ni}$ the Adams-Bashforth coefficients generated from the sequence of times \tilde{t}_n .

For computational purposes, it is useful to rewrite these steps as

$$\Delta \tilde{\mathbf{y}}_n = \sum_{q^1=m_n^1 - (k-1)}^{m_n^1} \cdots \sum_{q^S=m_n^S - (k-1)}^{m_n^S} \bar{I}_{n;q^1 \dots q^S} \mathbf{D}(\mathbf{y}_{q^1}^1, \dots, \mathbf{y}_{q^S}^S), \quad (2.17)$$

where the coefficient is

$$\bar{I}_{n;q^1 \dots q^S} = \Delta \tilde{t}_n \sum_{i=0}^{k-1} \tilde{\alpha}_{ni} \prod_{s=1}^S \ell_{m_n^s - q^s}(\tilde{t}_{n-i}; t_{m_n^s}^s, \dots, t_{m_n^s - (k-1)}^s). \quad (2.18)$$

The full change in the value of a given set of degrees of freedom over an entire step can then be obtained by summing the contributions of all these small steps. This will give for each set of degrees of freedom an equation of the form

$$\Delta \mathbf{y}_m^s = \Delta t_m^s \sum_{q^1} \cdots \sum_{q^S} a_{m;q^1 \dots q^S}^s \mathbf{D}(\mathbf{y}_{q^1}^1, \dots, \mathbf{y}_{q^S}^S) \quad (2.19)$$

for some coefficients $a_{m;q^1 \dots q^S}^s$.

²This freedom arises from the fact that the system of equations defining this interpolation is underdetermined for S and k greater than 1: we must find k^S fitting coefficients but there are only $\binom{k+S-1}{S}$ monomial terms of degree less than k (which are the ones relevant for an order k fit). A general choice of interpolation coefficients will result in an interpolating polynomial containing all terms of degree less than k in each of the t^s individually. We therefore have the freedom to modify the interpolation coefficients as long as the modification alters only terms in the interpolating polynomial of total degree at least k . This freedom could be used, for example, to set certain coefficients to zero to reduce the number of computations required or to decrease the effect of terms where the times on different sets have large mismatches.

In the case where the step size on each set is constant, the alternative sets of interpolation coefficients can be obtained by adding high-order products of discrete Chebyshev polynomials [25] to the coefficients in (2.16) or (2.18). In the general case we know of no simple method to calculate alternative coefficients. We have not investigated the use of such alternative coefficients in either of these cases.

$$\begin{array}{cccccc}
t^A \rightarrow \tilde{t} & \tilde{t} \rightarrow t^A & & & \tilde{t} \rightarrow t^B & t^B \rightarrow \tilde{t} \\
n_3^A = 4 & m_4^A = 3 & t_3^A \text{ --- } \tilde{t}_4 \text{ --- } & t_2^B & m_4^B = 2 & n_2^B = 4 \\
n_2^A = 3 & m_3^A = 2 & t_2^A \text{ --- } \tilde{t}_3 & & m_3^B = 1 & \\
n_1^A = 2 & m_2^A = 1 & t_1^A \text{ --- } \tilde{t}_2 & & m_2^B = 1 & \\
& m_1^A = 0 & & \tilde{t}_1 \text{ --- } t_1^B & m_1^B = 1 & n_1^B = 1 \\
n_0^A = 0 & m_0^A = 0 & t_0^A \text{ --- } \tilde{t}_0 \text{ --- } & t_0^B & m_0^B = 0 & n_0^B = 0
\end{array}$$

Figure 2.2: Example of the values of m_n^s and n_m^s for an arbitrarily chosen step pattern on two sets. These quantities give a mapping between the indices of the sequences of times t_m^s and \tilde{t}_n , with n_m^s mapping indices of t_m^s to the corresponding indices of \tilde{t}_n and m_n^s performing the reverse map. In cases where there is no t_m^s corresponding to a given \tilde{t}_n the index given by m_n^s is for the most recent step.

2.3 Special cases

2.3.1 Element splitting

These equations involve many more evaluations of the derivative than the standard GTS Adams-Bashforth method, so in the form (2.19) the LTS method is unlikely to be more efficient. However, if the couplings between the sets of degrees of freedom are inexpensive to calculate compared to the interactions within each set, then the required number of evaluations can be reduced.. Let us suppose that the derivative on set s is split into a “volume” portion only depending on set s itself and a “boundary” portion encoding the coupling to other sets:

$$\mathbf{D}^s(\mathbf{y}_{q^1}^1, \dots, \mathbf{y}_{q^s}^s) = \mathbf{V}^s(\mathbf{y}_{q^s}^s) + \mathbf{B}^s(\mathbf{y}_{q^1}^1, \dots, \mathbf{y}_{q^s}^s). \quad (2.20)$$

These names are motivated by finite volume and discontinuous Galerkin methods, where the terms from the interior and boundaries of elements split in this manner. Substituting this into (2.17) and summing over the small steps, the volume contribution to the full step on set s is

$$(\Delta \mathbf{y}_m^s)_{\text{vol}} = \sum_{q^s=m-(k-1)}^m \left[\sum_{n=n_m^s}^{n_{m+1}^s-1} \sum_{q^1=m_n^1-(k-1)}^{m_n^1} \cdots \sum_{/q^s} \cdots \sum_{q^s=m_n^s-(k-1)}^{m_n^s} \bar{I}_{n;q^1 \dots q^s} \right] \mathbf{V}^s(\mathbf{y}_{q^s}^s), \quad (2.21)$$

where n_m^s is defined by $\tilde{t}_{n_m^s} = t_m^s$ (see Figure 2.2). This is the same form as the GTS Adams-Bashforth method (2.2) using the bracketed expression as coefficients (absorbing the Δt factor). The bracketed expression does not depend on the form of the derivative, so to evaluate it we can take the boundary coupling \mathbf{B}^s to be zero, in which case this is the only contribution to the step. As this is then a k th-order GTS method and the Adams-Bashforth method is the unique k th-order method of this form, the bracketed quantity must be the standard Adams-Bashforth coefficient. Returning to the general case with a coupling, this shows that a set of degrees of freedom can be evolved using the standard Adams-Bashforth method for the volume portion with only the coupling terms evaluated using (2.17).

This simplification applies in intermediate cases as well: if the full derivative can be split into portions each of which depends on only some of the degree-of-freedom sets, each of those contributions to the step can be calculated independently using (2.17) ignoring non-contributing sets. In calculations where the sets are only coupled pairwise, this implies that only the $S = 2$ case need be considered.

2.3.2 Two-set case

In the common case where the sets of degrees of freedom are only coupled pairwise the update method reduces to a collection of standard Adams-Bashforth methods and LTS methods with $S = 2$. For the two-set case, we call the sets A and B and define the selection functions Θ_n^A , Θ_n^B , and Θ_n^{AB} to be one if \tilde{t}_n is an evaluation time for only set A , only set B , or both sets, respectively. By construction, a time evaluated on neither set can never occur. These selection functions sum to one, so

we can write $\bar{I}_{n;q^A q^B} = \bar{I}_{n;q^A q^B}^A + \bar{I}_{n;q^A q^B}^B + \bar{I}_{n;q^A q^B}^{AB}$ with

$$\bar{I}_{n;q^A q^B}^{A,B,AB} = \Delta \tilde{t}_n \sum_{i=0}^{k-1} \tilde{\alpha}_{ni} \Theta_{n-i}^{A,B,AB} \ell_{m_n^A - q^A}(\tilde{t}_{n-i}; t_{m_n^A}^A, \dots, t_{m_n^A - (k-1)}^A) \ell_{m_n^B - q^B}(\tilde{t}_{n-i}; t_{m_n^B}^B, \dots, t_{m_n^B - (k-1)}^B). \quad (2.22)$$

By the definition of m_n , \tilde{t}_n is not older than $t_{m_n^{A,B}}^{A,B}$, so, from the construction of the \tilde{t}_n we see that $\tilde{t}_{n-i} \geq t_{m_n^{A,B} - (k-1)}^{A,B}$. This implies that if the \tilde{t}_{n-i} is an evaluation time for either set, it is one of the control points in the corresponding Lagrange polynomial. We can therefore collapse those polynomials to obtain

$$\bar{I}_{n;q^A q^B}^A = \Theta_{n^A}^A \Delta \tilde{t}_n \tilde{\alpha}_{n,n-n^A} \ell_{m_n^B - q^B}(t_{q^A}^A; t_{m_n^B}^B, \dots, t_{m_n^B - (k-1)}^B) \quad (2.23)$$

$$\bar{I}_{n;q^A q^B}^{AB} = \delta_{t_{q^A}^A, t_{q^B}^B} \Delta \tilde{t}_n \tilde{\alpha}_{n,n-n^A}, \quad (2.24)$$

and $\bar{I}_{n;q^A q^B}^B = \bar{I}_{n;q^B q^A}^A$. Some example values are shown in Tables 2.1 and 2.2. The meaning of, for example, the first entry for (a) in Table 2.1 is that in (2.19) the coefficient $a_{0,0,1}^A = 115/64$ (where we have chosen to number the steps starting from $t = 0$ so the step on set B at Δt^B is step 1), so the equation for this step begins

$$\mathbf{y}^A(\Delta t^A) - \mathbf{y}^A(0) = \Delta t^A \left[\frac{115}{64} \mathbf{D}^A [\mathbf{y}^A(0), \mathbf{y}^B(\Delta t^B)] + \dots \right]. \quad (2.25)$$

Similarly, the lower-left entry for (c) in Table 2.2 indicates that one term in the second small step is

$$\mathbf{y}^B(2\Delta t^B) - \mathbf{y}^B(\Delta t^B) = \Delta t^B \left[\frac{2}{3} \mathbf{D}^A [\mathbf{y}^A(-2\Delta t^A), \mathbf{y}^B(\Delta t^B)] + \dots \right]. \quad (2.26)$$

Additional tables of coefficients can be found in Appendix 2.B.

2.4 Numerical results

We tested this scheme on a set of field equations evaluated using discontinuous Galerkin methods. In a DG formulation, the domain of evolution is divided into

| | | | | | | | | |
|--------------|---|-----|----------------|----------------|-------------------|----------------|------------------|----------------|
| Δt^A | — | (c) | $2\Delta t^B$ | (a) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ |
| | | (a) | Δt^B | 0 | $\frac{115}{64}$ | $\frac{7}{24}$ | $-\frac{11}{64}$ | 0 |
| | | (b) | 0 | $-\Delta t^A$ | $-\frac{115}{96}$ | 0 | $-\frac{11}{32}$ | $\frac{5}{24}$ |
| | | | $-\Delta t^B$ | $-2\Delta t^A$ | $\frac{23}{64}$ | 0 | $\frac{11}{192}$ | 0 |
| | | | $-2\Delta t^B$ | 0 | $\frac{23}{12}$ | $-\frac{1}{2}$ | 0 | 0 |
| | | | | $-\Delta t^A$ | 0 | -1 | $\frac{5}{12}$ | 0 |
| | | | | $-2\Delta t^A$ | 0 | $\frac{1}{6}$ | 0 | 0 |
| | | | | 0 | $\frac{115}{32}$ | $-\frac{4}{3}$ | $\frac{5}{32}$ | 0 |
| | | | | $-\Delta t^A$ | $-\frac{115}{48}$ | 0 | $\frac{5}{16}$ | 0 |
| | | | | $-2\Delta t^A$ | $\frac{23}{32}$ | 0 | $-\frac{5}{96}$ | 0 |

Table 2.1: A third-order method for two sets A and B with B evaluated twice as often as A . Coefficients for the derivatives in (2.19) evaluated using data from A and B at the times indicated for (a) a step of set A from 0 to Δt^A , and steps of set B (b) from 0 to Δt^B and (c) from Δt^B to $2\Delta t^B = \Delta t^A$.

elements, with each element containing a collection of nodes. The evolution equations are evaluated locally within each element and this collection of partial solutions is coupled by adding additional terms at the element boundaries obtained from comparison with neighboring elements. The application of the LTS scheme to this type of problem is natural: the elements themselves can each be evolved uniformly using a standard GTS method, with only the couplings to the neighbors using the more complicated LTS equations.

For our test problem, we evolved the scalar wave equation:

$$\frac{\partial^2 \psi}{\partial t^2} = \nabla^2 \psi. \tag{2.27}$$

For numerical purposes, this is usually written in a form that contains only first order temporal and spatial derivatives. This is done by introducing the additional

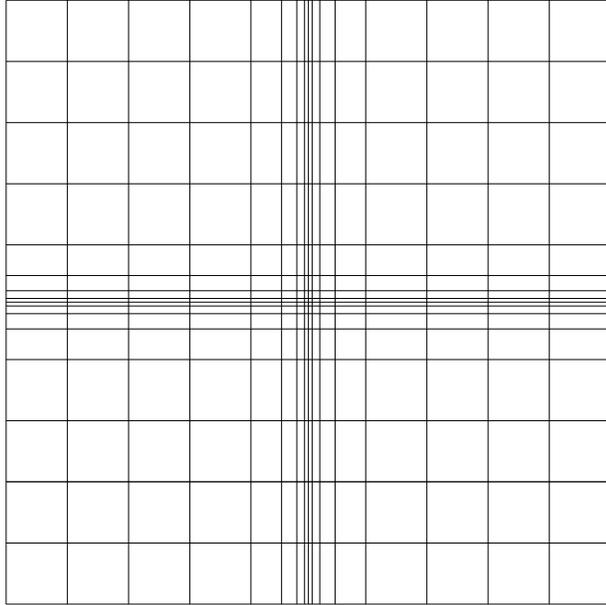


Figure 2.3: The domain used in the numerical tests: a square with periodic boundary conditions. The element pattern is symmetrical, with each half of each axis divided into four equal-sized segments and four smaller segments, each of which is half the size of its larger neighbor.

section 6 of [26]). For this system of PDEs, the integrals of π and $\vec{\Phi}$ are linear conserved quantities. This carries over to the discretized system as long as the discretization procedure preserves the standard vector calculus identities, which the DG scheme does. The integral of the energy density of the field, $E = (\pi^2 + \vec{\Phi}^2)/2$ is also an analytically conserved quantity, but it is not linear in the evolved variables.

We used for a domain a periodic two-dimensional square divided nonuniformly into rectangular elements, as shown in Figure 2.3. The largest elements are 16 times as large (linearly) as the smallest elements. The nodes in each element are distributed as Legendre-Gauss-Lobatto points in each dimension.

For our test solution we used a sinusoidal plane wave propagating diagonally across the square, with wavelength equal to half the length of the diagonal. Step sizes were restricted to be binary fractions of the wave period. When step sizes

were allowed to vary, they were chosen according to the CFL condition with the restriction that they must be binary fractions of the wave period. The step size in each element is determined by its smallest dimension, so all elements along the center of the refined cross take steps of the same size, 16 times as small as the steps on the largest elements. The step sizes then increased by factors of two moving outward to each next layer of elements.

As shown in Figure 2.4, the overall error in the evolution is larger when using an LTS integrator than when using a GTS integrator of the same smallest step size. This is because the GTS integration is taking unnecessarily small steps in the large elements, while the LTS integration is providing the largest time steps consistent with stability. If desired, the LTS error could be reduced by choosing steps using a criterion other than just stability. As the time steps throughout the domain are decreased, the numerical results converge to the analytic value at the expected rate, as shown for several integrator orders in Figure 2.5. Even for the largest possible time steps the errors in the linear conserved quantities are at roundoff level.

Switching from a standard third-order GTS scheme to the LTS scheme, while still using uniform step sizes throughout the domain, incurs a performance penalty of a factor of approximately 0.8 because of the increased computational cost of the boundary computations. However, once the step sizes are permitted to vary across the domain the total number of steps can be reduced by the ratio of the total number of steps across all elements during GTS and LTS, which in this case is $512/211$, providing a theoretical improvement by a factor of 2.43. This provides an upper bound on the speed increase obtainable for this problem by any LTS scheme. In practice, we observe an approximate speedup of 2.1 relative to the uniform-step LTS integration, leading to a speedup of approximately 1.6 relative

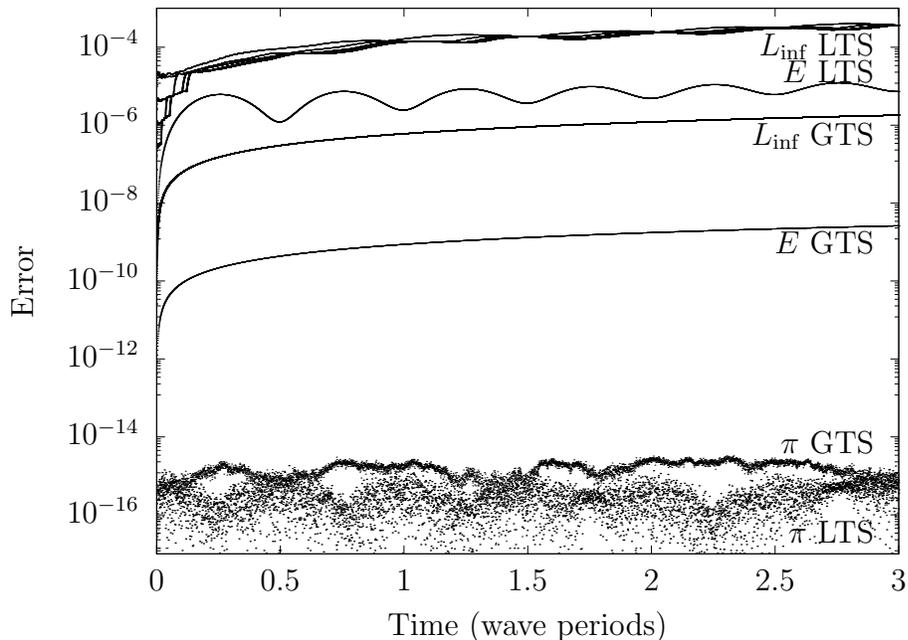


Figure 2.4: The L_{inf} norm of the error in ψ , π , and $\vec{\Phi}$, as well as the error in the integral over the domain of the conjugate momentum and energy density, for GTS and LTS evolutions. Both runs used a second-order stepper with 9^2 points per element. The integral of the conjugate momentum, being a linear conserved quantity, is constant to numerical roundoff for both methods, while the other errors reflect the integrator truncation error. The LTS truncation errors are larger because the error is dominated by the regions where the step size is large. The appearance of the LTS error as multiple lines is because the measured error is smaller at times when larger elements are not evaluated. The integrated quantities are only evaluated at times when all elements have data.

to the GTS scheme. The effects on integration speed from the LTS method are not strongly dependent on resolution, as shown in Figure 2.6, but integration is slower at higher time-stepper order, as shown in Figure 2.7, which is consistent with the need to evaluate more points for interpolating the couplings. The performance penalty of using an LTS scheme with a fixed step size does not vary significantly across the tested cases, which is expected because in the equal-step-size case no interpolation is necessary to compute couplings between elements.

All of these performance measurements are system-dependent, and should be improved with more complex systems than the scalar wave. The overhead from

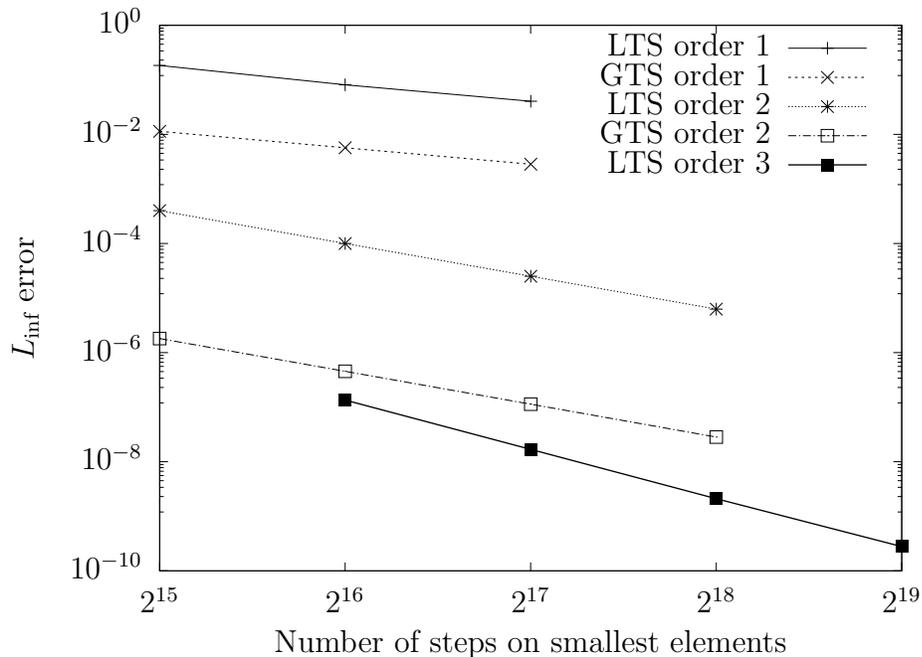


Figure 2.5: Maximum difference from the analytic solution over all grid points over the first three oscillations of the wave when artificially decreasing the step size below the CFL limit, showing the expected convergence rates. All simulations used 9^2 points per element. The third-order GTS errors (not shown) are dominated by the spatial discretization error of approximately 10^{-10} .

switching to LTS is independent of the system, and so should have a much smaller relative effect for systems with expensive right-hand sides. The overhead from increasing the integration order is proportional to the cost of the calculations on the element boundaries and so should decrease when these are much smaller than the right-hand-side cost.

2.5 Conclusions

When integrating systems of coupled ODEs, particularly those arising from discretizations of PDE systems, it is often the case that time-step-related instabilities arise primarily in a small subset of the variables being integrated. Using standard evolution schemes, this forces all degrees of freedom to be evolved with the most

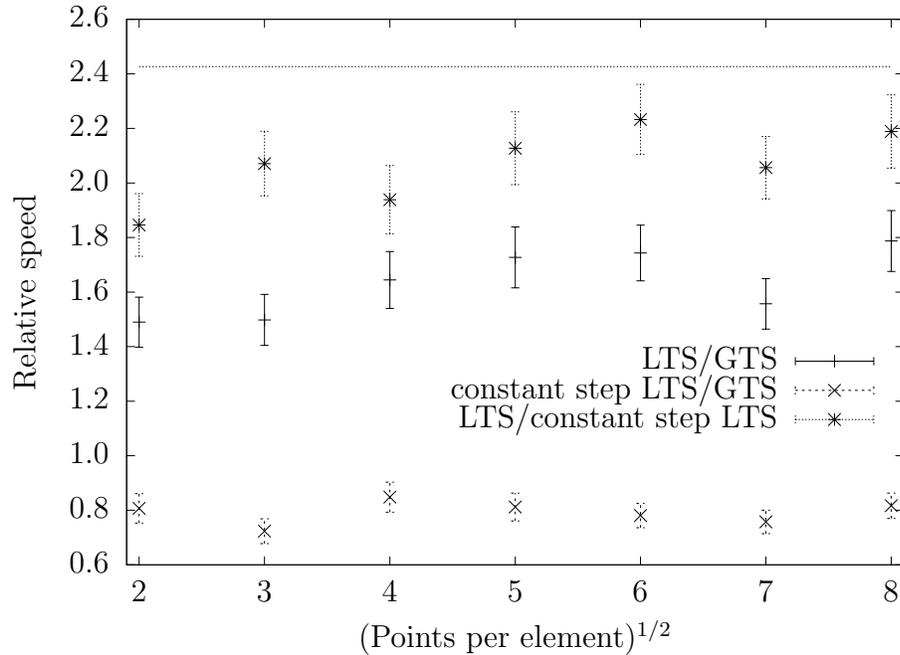


Figure 2.6: Comparisons of run speed using different third-order integrators at various element resolutions. The LTS and GTS algorithms are compared, and also compared to the LTS algorithm running with a constant global step size. The horizontal line shows the ratio of the number of steps on all elements taken when stepping globally and locally.

restrictive stable time step, potentially causing significant waste of computational resources. A local time-stepping integrator removes this requirement, allowing different degrees of freedom to be updated at different frequencies.

This paper has presented a local time-stepping scheme based on the Adams-Bashforth family of multistep integrators. This method allows arbitrary step choices, with a completely independent choice of time step for each variable. Unlike some previous schemes, it retains the full convergence order of the Adams-Bashforth integrator it is based on. This method is also conservative in that all linear conserved quantities of the system are constant to numerical roundoff under evolution.

The use of this method was demonstrated on a scalar wave equation evolved using the DG framework. The roundoff-level conservation of a conserved quantity

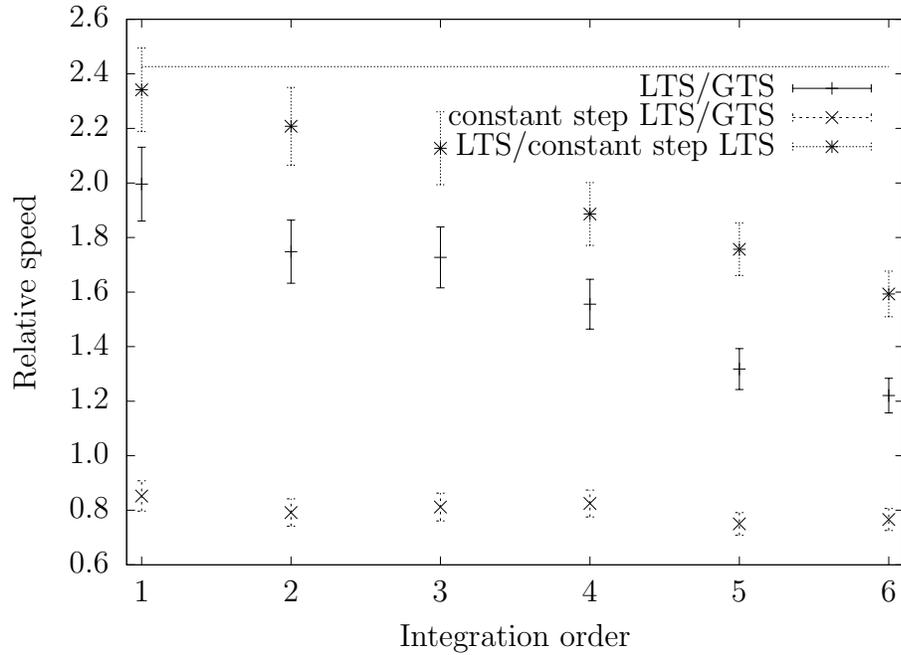


Figure 2.7: Comparisons of run speed using different integration orders on a domain with 5^2 points per element. The LTS and GTS algorithms are compared, and also compared to the LTS algorithm running with a constant global step size. The horizontal line shows the ratio of the number of steps on all elements taken when stepping globally and locally.

was demonstrated, and the expected convergence rate for other quantities was observed for multiple integrator orders. For this problem, we observe an evolution speed improvement by a factor of approximately 1.6 from switching from the global to the local scheme, although this number is strongly dependent on the integration order. We also expect a bigger speedup if the right-hand side of the equations is more complicated than the simple wave equation we used as a test case.

This method will be used for DG evolutions of general relativity and magneto-hydrodynamics in upcoming work using the SpECTRE code [27].

2.A Element splitting for general methods

When comparing integrators, one may wish to use a GTS integrator that is not usually expressed in terms of volume terms and boundary couplings (for example, a Runge-Kutta method) in a framework designed for an LTS integrator that is so expressed. This is easiest if the GTS integrator can be cast into the element splitting form (Section 2.3.1).

All common explicit GTS integrators (both multistep and substep) can be written in the form

$$u_{n+1} - u_n = \sum_i A_n^i (u_n - u_{n-i}) + \Delta t_n \sum_i B_n^i D(u_{n-i}). \quad (2.33)$$

Adams-Bashforth integrators are usually written in this form with the $A_n^i = 0$. Runge-Kutta methods take some manipulation. For example, the second-order midpoint method

$$u_{n+1} - u_n = \Delta t D \left[u_n + \frac{1}{2} \Delta t D(u_n) \right] \quad (2.34)$$

can be written as

$$u_{2n+1} - u_{2n} = \Delta t_{2n} D(u_{2n}) \quad (2.35)$$

$$u_{2n+2} - u_{2n+1} = -(u_{2n+1} - u_{2n}) + 2\Delta t_{2n+1} D(u_{2n+1}), \quad (2.36)$$

where we have renumbered the steps so that the even numbered ones are the results of complete RK steps and $\Delta t_n = \Delta t/2$.

For local time-stepping, the derivative values can depend on an additional set of values v_j (which have their own, similar, update equation), but where we still expect the update rule to have the form of a linear combination:

$$u_{n+1} - u_n = \sum_i A_n^i (u_n - u_{n-i}) + \Delta t_n \sum_{i,j} B_n^{ij} D(u_{n-i}, v_{m_n-j}). \quad (2.37)$$

We now perform an element splitting as in Section 2.3.1 by writing $D(u, v) = V(u) + B(u, v)$. Substituting this in gives

$$u_{n+1} - u_n = \left[\sum_i A_n^i (u_n - u_{n-i}) + \Delta t_n \sum_i \left(\sum_j B_n^{ij} \right) V(u_{n-i}) \right] + \Delta t_n \sum_{i,j} B_n^{ij} B(u_{n-i}, v_{m_n-j}). \quad (2.38)$$

Since a general method must be independent of the details of the V and B functions, the bracketed terms in (2.38) must be the standard GTS method operating with only the “volume” portion of the equations, and the last term is a coupling correction. Notably, the coupling term does not require the function values directly, but only the value of the coupling evaluated at those values.

When using a GTS method in an LTS framework, the u and v will be evaluated at the same sequence of times and the coefficients B_n^{ij} will be diagonal in i, j . Comparing (2.33) to the bracketed term in (2.38), we see that $\sum_j B_n^{ij} = B_n^i$, so for a GTS method $B_n^{ij} = \delta_{ij} B_n^i$. Combining all this, we find that

$$u_{n+1} - u_n = \left[\sum_i A_n^i (u_n - u_{n-i}) + \Delta t_n \sum_i B_n^i V(u_{n-i}) \right] + \Delta t_n \sum_i B_n^i B(u_{n-i}, v_{n-i}), \quad (2.39)$$

that is, when using an arbitrary GTS integrator in a framework designed for LTS, one can evaluate the volume term using the standard GTS rule and the coupling contribution by using the usual update formula but with all the non-derivative terms set to zero. For the midpoint Runge-Kutta scheme above, this gives the split rule

$$u_{2n+1} - u_{2n} = \Delta t_{2n} V(u_{2n}) + \Delta t_{2n} B(u_{2n}, v_{2n}) \quad (2.40)$$

$$u_{2n+2} - u_{2n+1} = \left[(u_{2n+1} - u_{2n+1-1}) + \Delta t_{2n+1} V(u_{2n+1}) \right] + \Delta t_{2n+1} B(u_{2n+1}, v_{2n+1}). \quad (2.41)$$

2.B Tables of coefficients for 2 : 1 LTS rules

Below are tables of coefficients for order 2, 3, and 4 LTS rules with 2 : 1 stepping, as well as the coefficients for transitioning between LTS and GTS stepping in these cases. The step patterns corresponding to these tables are shown in Figure 2.8.

For the transition rules, the number of steps requiring special coefficients depends on the order of the integrator. Only tables for steps affected by the transition are shown below, after which either the 2 : 1 rule or the GTS rule should be used, as appropriate.

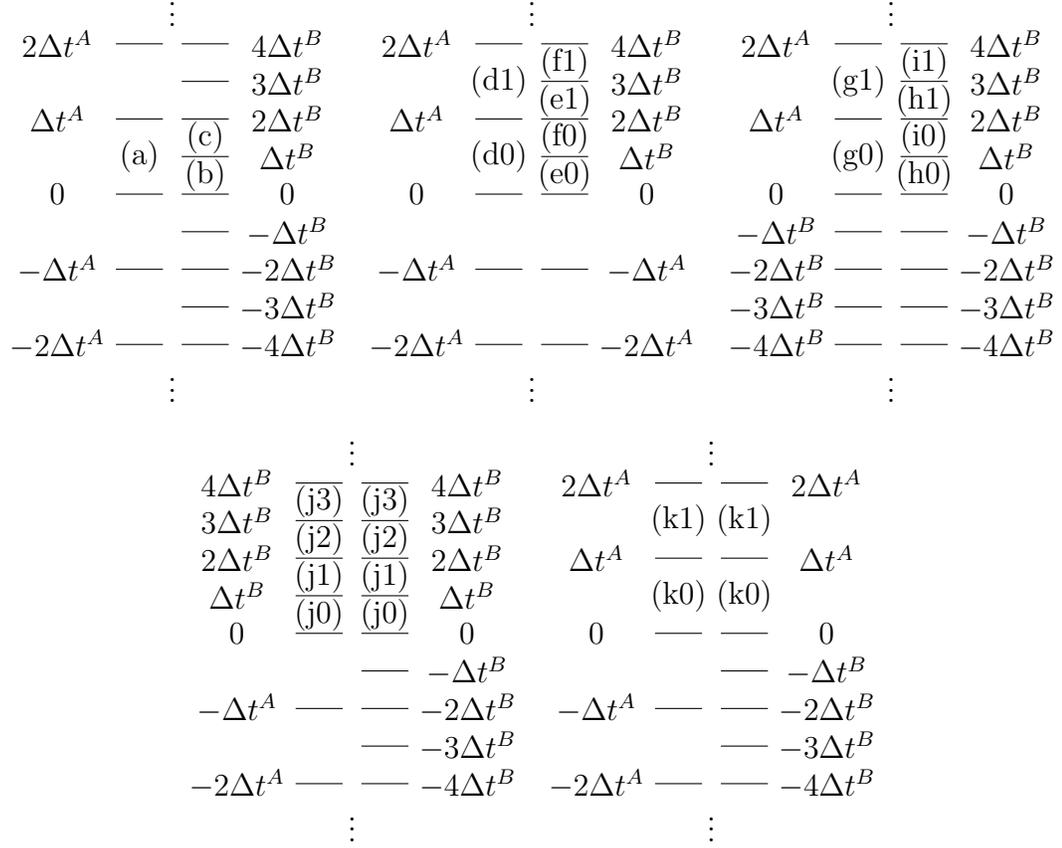


Figure 2.8: Step patterns during (a–c) steady state 2 : 1 evolution, (d–f) transition to LTS by decreasing a step size, (g–i) transition to LTS by increasing a step size, (j) transition back to GTS by decreasing a step size, and (k) transition back to GTS by increasing a step size. The labels correspond to the tables given in Appendix 2.B. The coefficients for transitioning back to GTS are the same for both elements. The numbered labels are extended upwards as necessary until the steady-state values are reached.

2.B.1 Order 2

LTS 2 : 1 rule

| | | | | | | | | | |
|---------------|----------------|---------------|----------------|---------------|---------------|----------------|---------------|----------------|----------------|
| (a) | Δt^B | 0 | $-\Delta t^B$ | (b) | 0 | $-\Delta t^B$ | (c) | Δt^B | 0 |
| 0 | $\frac{9}{8}$ | $\frac{1}{2}$ | $-\frac{1}{8}$ | 0 | $\frac{3}{2}$ | $-\frac{1}{4}$ | 0 | $\frac{9}{4}$ | $-\frac{1}{2}$ |
| $-\Delta t^A$ | $-\frac{3}{8}$ | 0 | $-\frac{1}{8}$ | $-\Delta t^A$ | 0 | $-\frac{1}{4}$ | $-\Delta t^A$ | $-\frac{3}{4}$ | 0 |

Transition to LTS by decreasing a step size

| | | | | | | | | | |
|---------------|----------------|---------------|----------------|---------------|---------------|----------------|---------------|----------------|----------------|
| (d0) | Δt^B | 0 | $-\Delta t^A$ | (e0) | 0 | $-\Delta t^A$ | (f0) | Δt^B | 0 |
| 0 | $\frac{9}{8}$ | $\frac{3}{8}$ | 0 | 0 | $\frac{5}{4}$ | 0 | 0 | $\frac{9}{4}$ | $-\frac{1}{2}$ |
| $-\Delta t^A$ | $-\frac{3}{8}$ | 0 | $-\frac{1}{8}$ | $-\Delta t^A$ | 0 | $-\frac{1}{4}$ | $-\Delta t^A$ | $-\frac{3}{4}$ | 0 |

Transition to LTS by increasing a step size

| | | | | | | | | | |
|---------------|----------------|---------------|----------------|---------------|---------------|----------------|---------------|----------------|----------------|
| (g0) | Δt^B | 0 | $-\Delta t^B$ | (h0) | 0 | $-\Delta t^B$ | (i0) | Δt^B | 0 |
| 0 | $\frac{3}{2}$ | $\frac{1}{2}$ | 0 | 0 | $\frac{3}{2}$ | 0 | 0 | 3 | $-\frac{1}{2}$ |
| $-\Delta t^B$ | $-\frac{3}{4}$ | 0 | $-\frac{1}{4}$ | $-\Delta t^B$ | 0 | $-\frac{1}{2}$ | $-\Delta t^B$ | $-\frac{3}{2}$ | 0 |

Transitioning to GTS

| | | | | | |
|---------------|---------------|----------------|---------------|---|----------------|
| (j0) | 0 | $-\Delta t^B$ | (k0) | 0 | $-\Delta t^B$ |
| 0 | $\frac{3}{2}$ | $-\frac{1}{4}$ | 0 | 2 | $-\frac{1}{2}$ |
| $-\Delta t^A$ | 0 | $-\frac{1}{4}$ | $-\Delta t^A$ | 0 | $-\frac{1}{2}$ |

2.B.2 Order 3

LTS 2 : 1 rule

| (a) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | (b) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ |
|----------------|-------------------|----------------|------------------|----------------|----------------|-----------------|----------------|----------------|
| 0 | $\frac{115}{64}$ | $\frac{7}{24}$ | $-\frac{11}{64}$ | 0 | 0 | $\frac{23}{12}$ | $-\frac{1}{2}$ | 0 |
| $-\Delta t^A$ | $-\frac{115}{96}$ | 0 | $-\frac{11}{32}$ | $\frac{5}{24}$ | $-\Delta t^A$ | 0 | -1 | $\frac{5}{12}$ |
| $-2\Delta t^A$ | $\frac{23}{64}$ | 0 | $\frac{11}{192}$ | 0 | $-2\Delta t^A$ | 0 | $\frac{1}{6}$ | 0 |

| (c) | Δt^B | 0 | $-\Delta t^B$ |
|----------------|-------------------|----------------|-----------------|
| 0 | $\frac{115}{32}$ | $-\frac{4}{3}$ | $\frac{5}{32}$ |
| $-\Delta t^A$ | $-\frac{115}{48}$ | 0 | $\frac{5}{16}$ |
| $-2\Delta t^A$ | $\frac{23}{32}$ | 0 | $-\frac{5}{96}$ |

Transition to LTS by decreasing a step size

| (d0) | Δt^B | 0 | $-\Delta t^A$ | $-2\Delta t^A$ | (e0) | 0 | $-\Delta t^A$ | $-2\Delta t^A$ |
|----------------|-----------------|---------------|----------------|----------------|----------------|-----------------|-----------------|----------------|
| 0 | $\frac{5}{3}$ | $\frac{1}{4}$ | 0 | 0 | 0 | $\frac{17}{12}$ | 0 | 0 |
| $-\Delta t^A$ | $-\frac{10}{9}$ | 0 | $-\frac{2}{9}$ | 0 | $-\Delta t^A$ | 0 | $-\frac{7}{12}$ | 0 |
| $-2\Delta t^A$ | $\frac{1}{3}$ | 0 | 0 | $\frac{1}{12}$ | $-2\Delta t^A$ | 0 | 0 | $\frac{1}{6}$ |

| (f0) | Δt^B | 0 | $-\Delta t^A$ |
|----------------|-----------------|------------------|----------------|
| 0 | $\frac{10}{3}$ | $-\frac{11}{12}$ | 0 |
| $-\Delta t^A$ | $-\frac{20}{9}$ | 0 | $\frac{5}{36}$ |
| $-2\Delta t^A$ | $\frac{2}{3}$ | 0 | 0 |

Transition to LTS by increasing a step size

| | | | | | | | | | |
|----------------|-----------------|----------------|------------------|----------------|----------------|------------------|----------------|------------------|----------------|
| (g0) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | (h0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | |
| 0 | $\frac{23}{8}$ | $\frac{7}{24}$ | 0 | 0 | 0 | $\frac{23}{12}$ | 0 | 0 | |
| $-\Delta t^B$ | $-\frac{23}{8}$ | 0 | $-\frac{11}{24}$ | 0 | $-\Delta t^B$ | 0 | $-\frac{4}{3}$ | 0 | |
| $-2\Delta t^B$ | $\frac{23}{24}$ | 0 | 0 | $\frac{5}{24}$ | $-2\Delta t^B$ | 0 | 0 | $\frac{5}{12}$ | |
| (i0) | Δt^B | 0 | $-\Delta t^B$ | | (g1) | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | 0 |
| 0 | $\frac{23}{4}$ | $-\frac{4}{3}$ | 0 | | Δt^A | $\frac{23}{12}$ | $\frac{7}{24}$ | $-\frac{11}{72}$ | 0 |
| $-\Delta t^B$ | $-\frac{23}{4}$ | 0 | $\frac{5}{12}$ | | 0 | $-\frac{23}{12}$ | 0 | $-\frac{11}{24}$ | $\frac{5}{24}$ |
| $-2\Delta t^B$ | $\frac{23}{12}$ | 0 | 0 | | $-\Delta t^B$ | $\frac{23}{24}$ | 0 | $\frac{11}{72}$ | 0 |
| (h1) | $2\Delta t^B$ | Δt^B | 0 | | (i1) | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | |
| Δt^A | $\frac{23}{12}$ | $-\frac{4}{9}$ | 0 | | Δt^A | $\frac{23}{6}$ | $-\frac{4}{3}$ | $\frac{5}{36}$ | |
| 0 | 0 | $-\frac{4}{3}$ | $\frac{5}{12}$ | | 0 | $-\frac{23}{6}$ | 0 | $\frac{5}{12}$ | |
| $-\Delta t^B$ | 0 | $\frac{4}{9}$ | 0 | | $-\Delta t^B$ | $\frac{23}{12}$ | 0 | $-\frac{5}{36}$ | |

Transitioning to GTS by decreasing a step size

| | | | | | | | |
|----------------|-----------------|----------------|----------------|---------------|-----------------|----------------|-----------------|
| (j0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | (j1) | Δt^B | 0 | $-\Delta t^B$ |
| 0 | $\frac{23}{12}$ | $-\frac{1}{2}$ | 0 | Δt^B | $\frac{23}{12}$ | 0 | $-\frac{5}{36}$ |
| $-\Delta t^A$ | 0 | -1 | $\frac{5}{12}$ | 0 | 0 | $-\frac{4}{3}$ | $\frac{5}{12}$ |
| $-2\Delta t^A$ | 0 | $\frac{1}{6}$ | 0 | $-\Delta t^A$ | 0 | 0 | $\frac{5}{36}$ |

Transitioning to GTS by increasing a step size

| (k0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | (k1) | Δt^A | 0 | $-\Delta t^B$ |
|----------------|----------------|----------------|----------------|---------------|-----------------|-----------------|-----------------|
| 0 | $\frac{19}{6}$ | $-\frac{5}{4}$ | 0 | Δt^A | $\frac{37}{18}$ | 0 | $-\frac{5}{36}$ |
| $-\Delta t^A$ | 0 | $-\frac{5}{2}$ | $\frac{7}{6}$ | 0 | 0 | $-\frac{13}{6}$ | $\frac{5}{6}$ |
| $-2\Delta t^A$ | 0 | $\frac{5}{12}$ | 0 | $-\Delta t^A$ | 0 | 0 | $\frac{5}{12}$ |

2.B.3 Order 4

LTS 2 : 1 rule

| (a) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ | (b) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ |
|----------------|---------------------|-----------------|-------------------|----------------|-------------------|----------------|-----------------|--------------------|-----------------|-------------------|
| 0 | $\frac{1925}{768}$ | $-\frac{1}{12}$ | $-\frac{55}{384}$ | 0 | $\frac{3}{256}$ | 0 | $\frac{55}{24}$ | $-\frac{295}{384}$ | 0 | $\frac{3}{128}$ |
| $-\Delta t^A$ | $-\frac{1925}{768}$ | 0 | $-\frac{55}{128}$ | $\frac{7}{12}$ | $-\frac{27}{256}$ | $-\Delta t^A$ | 0 | $-\frac{295}{128}$ | $\frac{37}{24}$ | $-\frac{27}{128}$ |
| $-2\Delta t^A$ | $\frac{385}{256}$ | 0 | $\frac{55}{384}$ | 0 | $-\frac{27}{256}$ | $-2\Delta t^A$ | 0 | $\frac{295}{384}$ | 0 | $-\frac{27}{128}$ |
| $-3\Delta t^A$ | $-\frac{275}{768}$ | 0 | $-\frac{11}{384}$ | 0 | $\frac{3}{256}$ | $-3\Delta t^A$ | 0 | $-\frac{59}{384}$ | 0 | $\frac{3}{128}$ |

| (c) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ |
|----------------|---------------------|------------------|--------------------|----------------|
| 0 | $\frac{1925}{384}$ | $-\frac{59}{24}$ | $\frac{185}{384}$ | 0 |
| $-\Delta t^A$ | $-\frac{1925}{384}$ | 0 | $\frac{185}{128}$ | $-\frac{3}{8}$ |
| $-2\Delta t^A$ | $\frac{385}{128}$ | 0 | $-\frac{185}{384}$ | 0 |
| $-3\Delta t^A$ | $-\frac{275}{384}$ | 0 | $\frac{37}{384}$ | 0 |

Transition to LTS by decreasing a step size

| | | | | | | | | | | |
|----------------|--------------------|--------------------|-------------------|--------------------|-------------------|---------------------|-------------------|--------------------|-------------------|-------------------|
| (d0) | Δt^B | 0 | $-\Delta t^A$ | $-2\Delta t^A$ | $-3\Delta t^A$ | (e0) | 0 | $-\Delta t^A$ | $-2\Delta t^A$ | $-3\Delta t^A$ |
| 0 | $\frac{833}{384}$ | $\frac{47}{384}$ | 0 | 0 | 0 | 0 | $\frac{99}{64}$ | 0 | 0 | 0 |
| $-\Delta t^A$ | $-\frac{833}{384}$ | 0 | $-\frac{37}{128}$ | 0 | 0 | $-\Delta t^A$ | 0 | $-\frac{187}{192}$ | 0 | 0 |
| $-2\Delta t^A$ | $\frac{833}{640}$ | 0 | 0 | $\frac{461}{1920}$ | 0 | $-2\Delta t^A$ | 0 | 0 | $\frac{107}{192}$ | 0 |
| $-3\Delta t^A$ | $-\frac{119}{384}$ | 0 | 0 | 0 | $-\frac{25}{384}$ | $-3\Delta t^A$ | 0 | 0 | 0 | $-\frac{25}{192}$ |
| (f0) | Δt^B | 0 | $-\Delta t^A$ | $-2\Delta t^A$ | | | | | | |
| 0 | $\frac{833}{192}$ | $-\frac{125}{96}$ | 0 | 0 | (d1) | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | 0 | $-\Delta t^A$ |
| $-\Delta t^A$ | $-\frac{833}{192}$ | 0 | $\frac{19}{48}$ | 0 | Δt^A | $\frac{1925}{768}$ | $-\frac{25}{192}$ | $-\frac{65}{768}$ | 0 | 0 |
| $-2\Delta t^A$ | $\frac{833}{320}$ | 0 | 0 | $-\frac{37}{480}$ | 0 | $-\frac{1925}{768}$ | 0 | $-\frac{65}{256}$ | $\frac{29}{96}$ | 0 |
| $-3\Delta t^A$ | $-\frac{119}{192}$ | 0 | 0 | 0 | $-\Delta t^A$ | $\frac{385}{256}$ | 0 | $\frac{65}{768}$ | 0 | $-\frac{3}{64}$ |
| (e1) | $2\Delta t^B$ | Δt^B | 0 | $-\Delta t^A$ | $-2\Delta t^A$ | $-\frac{275}{768}$ | 0 | $-\frac{13}{768}$ | 0 | 0 |
| Δt^A | $\frac{211}{96}$ | $-\frac{125}{192}$ | 0 | 0 | (f1) | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | 0 | |
| 0 | 0 | $-\frac{125}{64}$ | $\frac{47}{48}$ | 0 | Δt^A | $\frac{1925}{384}$ | $-\frac{59}{24}$ | $\frac{185}{384}$ | 0 | |
| $-\Delta t^A$ | 0 | $\frac{125}{192}$ | 0 | $-\frac{3}{32}$ | 0 | $-\frac{1925}{384}$ | 0 | $\frac{185}{128}$ | $-\frac{3}{8}$ | |
| $-2\Delta t^A$ | 0 | $-\frac{25}{192}$ | 0 | 0 | $-\Delta t^A$ | $\frac{385}{128}$ | 0 | $-\frac{185}{384}$ | 0 | |
| | | | | | $-2\Delta t^A$ | $-\frac{275}{384}$ | 0 | $\frac{37}{384}$ | 0 | |

Transition to LTS by increasing a step size

| (g0) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ | (h0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ | | | | | |
|----------------|-------------------|------------------|-------------------|----------------|-----------------|-------------------|------------------|-------------------|-----------------|-----------------|---------------|---------------|---------------|--------------|---------------|
| 0 | $\frac{55}{12}$ | $-\frac{1}{12}$ | 0 | 0 | 0 | 0 | $\frac{55}{24}$ | 0 | 0 | 0 | | | | | |
| $-\Delta t^B$ | $-\frac{55}{8}$ | 0 | $-\frac{11}{24}$ | 0 | 0 | $-\Delta t^B$ | 0 | $-\frac{59}{24}$ | 0 | 0 | | | | | |
| $-2\Delta t^B$ | $\frac{55}{12}$ | 0 | 0 | $\frac{7}{12}$ | 0 | $-2\Delta t^B$ | 0 | 0 | $\frac{37}{24}$ | 0 | | | | | |
| $-3\Delta t^B$ | $-\frac{55}{48}$ | 0 | 0 | 0 | $-\frac{3}{16}$ | $-3\Delta t^B$ | 0 | 0 | 0 | $-\frac{3}{8}$ | | | | | |
| (i0) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | (g1) | | | | | | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | 0 | $-\Delta t^B$ |
| 0 | $\frac{55}{6}$ | $-\frac{59}{24}$ | 0 | 0 | Δt^A | $\frac{275}{96}$ | $-\frac{1}{12}$ | $-\frac{11}{96}$ | 0 | 0 | | | | | |
| $-\Delta t^B$ | $-\frac{55}{4}$ | 0 | $\frac{37}{24}$ | 0 | 0 | $-\frac{275}{48}$ | 0 | $-\frac{11}{16}$ | $\frac{7}{12}$ | 0 | | | | | |
| $-2\Delta t^B$ | $\frac{55}{6}$ | 0 | 0 | $-\frac{3}{8}$ | $-\Delta t^B$ | $\frac{275}{48}$ | 0 | $\frac{11}{24}$ | 0 | $-\frac{3}{16}$ | | | | | |
| $-3\Delta t^B$ | $-\frac{55}{24}$ | 0 | 0 | 0 | $-2\Delta t^B$ | $-\frac{55}{32}$ | 0 | $-\frac{11}{96}$ | 0 | 0 | | | | | |
| (h1) | $2\Delta t^B$ | Δt^B | 0 | $-\Delta t^B$ | (i1) | | | | | | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | 0 | |
| Δt^A | $\frac{55}{24}$ | $-\frac{59}{96}$ | 0 | 0 | Δt^A | $\frac{275}{48}$ | $-\frac{59}{24}$ | $\frac{37}{96}$ | 0 | | | | | | |
| 0 | 0 | $-\frac{59}{16}$ | $\frac{37}{24}$ | 0 | 0 | $-\frac{275}{24}$ | 0 | $\frac{37}{16}$ | $-\frac{3}{8}$ | | | | | | |
| $-\Delta t^B$ | 0 | $\frac{59}{24}$ | 0 | $-\frac{3}{8}$ | $-\Delta t^B$ | $\frac{275}{24}$ | 0 | $-\frac{37}{24}$ | 0 | | | | | | |
| $-2\Delta t^B$ | 0 | $-\frac{59}{96}$ | 0 | 0 | $-2\Delta t^B$ | $-\frac{55}{16}$ | 0 | $\frac{37}{96}$ | 0 | | | | | | |
| (g2) | $5\Delta t^B$ | $4\Delta t^B$ | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | (h2) | | | | | $4\Delta t^B$ | $3\Delta t^B$ | $2\Delta t^B$ | Δt^B | |
| $2\Delta t^A$ | $\frac{165}{64}$ | $-\frac{1}{12}$ | $-\frac{11}{80}$ | 0 | $\frac{3}{320}$ | $2\Delta t^A$ | $\frac{55}{24}$ | $-\frac{59}{80}$ | 0 | $\frac{3}{160}$ | | | | | |
| Δt^A | $-\frac{275}{96}$ | 0 | $-\frac{11}{24}$ | $\frac{7}{12}$ | $-\frac{3}{32}$ | Δt^A | 0 | $-\frac{59}{24}$ | $\frac{37}{24}$ | $-\frac{3}{16}$ | | | | | |
| 0 | $\frac{165}{64}$ | 0 | $\frac{11}{48}$ | 0 | $-\frac{9}{64}$ | 0 | 0 | $\frac{59}{48}$ | 0 | $-\frac{9}{32}$ | | | | | |
| $-\Delta t^B$ | $-\frac{55}{48}$ | 0 | $-\frac{11}{120}$ | 0 | $\frac{3}{80}$ | $-\Delta t^B$ | 0 | $-\frac{59}{120}$ | 0 | $\frac{3}{40}$ | | | | | |

| (i2) | $5\Delta t^B$ | $4\Delta t^B$ | $3\Delta t^B$ | $2\Delta t^B$ |
|---------------|-------------------|------------------|------------------|----------------|
| $2\Delta t^A$ | $\frac{165}{32}$ | $-\frac{59}{24}$ | $\frac{37}{80}$ | 0 |
| Δt^A | $-\frac{275}{48}$ | 0 | $\frac{37}{24}$ | $-\frac{3}{8}$ |
| 0 | $\frac{165}{32}$ | 0 | $-\frac{37}{48}$ | 0 |
| $-\Delta t^B$ | $-\frac{55}{24}$ | 0 | $\frac{37}{120}$ | 0 |

Transitioning to GTS by decreasing a step size

| (j0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ | (j1) | Δt^B | 0 | $-\Delta t^B$ | $-2\Delta t^B$ |
|----------------|-----------------|--------------------|-----------------|-------------------|----------------|-----------------|------------------|-------------------|----------------|
| 0 | $\frac{55}{24}$ | $-\frac{295}{384}$ | 0 | $\frac{3}{128}$ | Δt^B | $\frac{55}{24}$ | 0 | $-\frac{37}{120}$ | 0 |
| $-\Delta t^A$ | 0 | $-\frac{295}{128}$ | $\frac{37}{24}$ | $-\frac{27}{128}$ | 0 | 0 | $-\frac{59}{24}$ | $\frac{37}{32}$ | 0 |
| $-2\Delta t^A$ | 0 | $\frac{295}{384}$ | 0 | $-\frac{27}{128}$ | $-\Delta t^A$ | 0 | 0 | $\frac{37}{48}$ | $-\frac{3}{8}$ |
| $-3\Delta t^A$ | 0 | $-\frac{59}{384}$ | 0 | $\frac{3}{128}$ | $-2\Delta t^A$ | 0 | 0 | $-\frac{37}{480}$ | 0 |

| (j2) | $2\Delta t^B$ | Δt^B | 0 | $-\Delta t^B$ |
|---------------|-----------------|------------------|-----------------|-----------------|
| $2\Delta t^B$ | $\frac{55}{24}$ | 0 | 0 | $-\frac{3}{32}$ |
| Δt^B | 0 | $-\frac{59}{24}$ | 0 | $\frac{3}{8}$ |
| 0 | 0 | 0 | $\frac{37}{24}$ | $-\frac{9}{16}$ |
| $-\Delta t^A$ | 0 | 0 | 0 | $-\frac{3}{32}$ |

Transitioning to GTS by increasing a step size

| (k0) | 0 | $-\Delta t^B$ | $-2\Delta t^B$ | $-3\Delta t^B$ | (k1) | Δt^A | 0 | $-\Delta t^B$ | $-2\Delta t^B$ |
|----------------|-----------------|------------------|----------------|-----------------|----------------|---------------|-----------------|----------------|-----------------|
| 0 | $\frac{9}{2}$ | $-\frac{55}{24}$ | 0 | $\frac{1}{12}$ | Δt^A | $\frac{8}{3}$ | 0 | $-\frac{3}{8}$ | 0 |
| $-\Delta t^A$ | 0 | $-\frac{55}{8}$ | $\frac{31}{6}$ | $-\frac{3}{4}$ | 0 | 0 | $-\frac{35}{6}$ | $\frac{27}{8}$ | 0 |
| $-2\Delta t^A$ | 0 | $\frac{55}{24}$ | 0 | $-\frac{3}{4}$ | $-\Delta t^A$ | 0 | 0 | $\frac{27}{8}$ | $-\frac{11}{6}$ |
| $-3\Delta t^A$ | 0 | $-\frac{11}{24}$ | 0 | $\frac{1}{12}$ | $-2\Delta t^A$ | 0 | 0 | $-\frac{3}{8}$ | 0 |
| (k2) | $2\Delta t^A$ | Δt^A | 0 | $-\Delta t^B$ | | | | | |
| $2\Delta t^A$ | $\frac{71}{30}$ | 0 | 0 | $-\frac{3}{40}$ | | | | | |
| Δt^A | 0 | $-\frac{17}{6}$ | 0 | $\frac{3}{8}$ | | | | | |
| 0 | 0 | 0 | $\frac{8}{3}$ | $-\frac{9}{8}$ | | | | | |
| $-\Delta t^A$ | 0 | 0 | 0 | $-\frac{3}{8}$ | | | | | |

**SPECTRE: A TASK-BASED DISCONTINUOUS GALERKIN CODE
FOR RELATIVISTIC ASTROPHYSICS¹**

3.1 Introduction

Many of the most energetic phenomena in the universe involve matter under extreme gravitational conditions. These phenomena include neutron-star binary mergers, accretion onto black holes, and supernova explosions. For many of these systems, the motion of this matter is expected to generate extremely strong magnetic fields. The matter and magnetic fields in these systems are governed by the equations of general relativistic magnetohydrodynamics (GRMHD). These equations admit a rich variety of solutions, which often include large-scale relativistic flows and small-scale phenomena such as shocks and turbulence.

Disks are a common feature in the neighborhood of all types of massive bodies. Around relativistic compact objects, they can form via accretion of surrounding matter (possibly captured from a companion star) or in the aftermath of a compact-object merger. In any case, the disk can be observed because of its very high temperature from the energy released by infalling matter. Disks may also power the jets observed originating in the vicinity of many black holes.

The dynamics of relativistic disk systems are not well understood. The radial motion of disk material is likely governed by physics at scales much smaller than the disk itself. The mechanism by which a disk drives a relativistic jet is also uncertain. As these systems are too small and distant for the relevant physics to be observed directly, answers to these questions are best found using large-scale numerical simulations. These simulations must capture physics at a wide range of

¹This chapter is based on a draft of a paper co-authored by members of the SXS collaboration that will be submitted for publication shortly.

length scales and run for at least the disk radial transport time.

The current generation of GRMHD codes is not accurate enough to provide useful predictions for most extreme systems. Increasing simulation resolutions can improve this, but is very computationally expensive. More appealing is using numerical methods with higher convergence orders, which can increase accuracy with significantly less cost than a similar improvement from resolution. Unfortunately, while higher-order methods handle smooth solutions very well, they are generally poor at handling discontinuities, such as fluid shocks, losing accuracy and sometimes failing completely.

Additionally, the time required to run simulations is already too long for most interesting astrophysical cases. The performance of individual computational processors has stagnated over the past decade, so to improve computational speed codes must be parallelized over more processors. New supercomputer clusters will soon routinely have millions of cores. Codes designed for running on thousands of processors generally scale poorly to massively parallel setups, however. As problems are divided up into an increasing numbers of parts, the amount of communication required during the simulation can become prohibitive, particularly for high-order methods.

Discontinuous Galerkin (DG) methods [28–33], together with a task-based parallelization strategy, have the potential to deal with these problems. DG methods offer high-order accuracy in smooth regions (although, for stability, increasing the scheme’s order requires decreasing the time step, which restricts the largest usable order in practice). They also offer robustness for shocks and other discontinuities. The methods are also well suited for parallelization: Their formulation in terms of local, non-overlapping elements requires only nearest-neighbor communication regardless of the scheme’s order of convergence. Additionally, these features allow

for comparatively straightforward *hp*-adaptivity and local time-stepping, enabling better load distribution across a large number of cores.

Despite extensive success in engineering and applied mathematics communities over the past two decades, applications of DG in relativity [34–38] and astrophysics [39–42] have typically been exploratory or confined to simple problems. Within the past year, however, there have been significant advances toward production codes for non-relativistic [43] and relativistic [26, 44] hydrodynamics, special relativistic magnetohydrodynamics [41, 45–49], and the Einstein equations [38, 50]. Most of these codes use MPI to implement a data parallelism strategy, though [46] uses task-based parallelism.

In this paper we describe the current state of the general relativity and astrophysics code SpECTRE, which was previously described in [27]. In the time since that publication, SpECTRE has been completely rewritten as an open-source code.² SpECTRE uses task-based parallelization to achieve good scaling of DG-based schemes to at least 100,000 processors.

This paper is organized as follows. Section 3.2 describes the formulation of GRMHD used in the problems presented here. Section 3.3 describes the algorithms used by SpECTRE to solve these equations. Results of the evolutions of a variety of GRMHD problems are presented in Section 3.4.

3.2 Equations

We adopt the standard 3+1 form of the spacetime metric, (see, e.g., [51, 52]),

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

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

²<https://github.com/sxs-collaboration/spectre>

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 $c = G = 1$.

SpECTRE currently solves equations in flux-balanced and first-order hyperbolic form. The general form of a flux-balanced conservation law in a curved spacetime is

$$\frac{1}{\sqrt{\gamma}}\partial_t(\sqrt{\gamma}U) + \frac{1}{\sqrt{\gamma}}\partial_i(\sqrt{\gamma}F^i) = S, \quad (3.2)$$

where $\gamma = \det(\gamma_{ij})$ is the determinant of the spatial metric, U is the state vector, F^i are the components of the flux vector, and S is the source vector.

We refer the reader to the literature [51, 53, 54] for a detailed description of the equations of general relativistic magnetohydrodynamics (GRMHD). If we ignore self-gravity, the GRMHD equations constitute a closed system that may be solved on a given background metric. We denote the rest-mass density of the fluid by ρ and its 4-velocity by u^a , where $u^a u_a = -1$. The dual of the Faraday tensor F^{ab} is

$$*F^{ab} = \frac{1}{2}\epsilon^{abcd}F_{cd}, \quad (3.3)$$

where ϵ^{abcd} is the Levi-Civita tensor. The equations governing the evolution of the GRMHD system are:

$$\nabla_a(\rho u^a) = 0 \quad (\text{rest-mass conservation}), \quad (3.4)$$

$$\nabla_a T^{ab} = 0 \quad (\text{energy-momentum conservation}), \quad (3.5)$$

$$\nabla_a *F^{ab} = 0 \quad (\text{homogeneous Maxwell equation}). \quad (3.6)$$

In the ideal MHD limit the stress tensor takes the form

$$T^{ab} = (\rho h)*u^a u^b + p*g^{ab} - b^a b^b \quad (3.7)$$

where

$$b^a = *F^{ab}u_b \quad (3.8)$$

is the magnetic field measured in the comoving frame of the fluid, and $(\rho h)^* = \rho h + b^2$ and $p^* = p + b^2/2$ are the enthalpy density and fluid pressure augmented by contributions of magnetic pressure $p_{\text{mag}} = b^2/2$, respectively.

We denote the unit normal vector to the spatial hypersurfaces as n^a , which is given by

$$n^a = (1/\alpha, -\beta^i/\alpha)^T, \quad (3.9)$$

$$n_a = (-\alpha, 0, 0, 0). \quad (3.10)$$

The spatial velocity of the fluid as measured by an observer at rest in the spatial hypersurfaces (“Eulerian observer”) is

$$v^i = \frac{1}{\alpha} \left(\frac{u^i}{u^0} + \beta^i \right), \quad (3.11)$$

with a corresponding Lorentz factor W given by

$$W = -u^a n_a = \alpha u^0 = \frac{1}{\sqrt{1 - \gamma_{ij} v^i v^j}}. \quad (3.12)$$

The electric and magnetic fields as measured by an Eulerian observer are given by

$$E^i = F^{ia} n_a = \alpha F^{0i}, \quad (3.13)$$

$$B^i = {}^*F^{ia} n_a = \alpha {}^*F^{0i}. \quad (3.14)$$

Finally, the comoving magnetic field b^a in terms of B^i is

$$b^0 = \frac{W}{\alpha} B^i v_i, \quad (3.15)$$

$$b^i = \frac{B^i + \alpha b^0 u^i}{W}, \quad (3.16)$$

while $b^2 = b^a b_a$ is given by

$$b^2 = \frac{B^2}{W^2} + (B^i v_i)^2. \quad (3.17)$$

We now recast the GRMHD equations in a 3+1 split by projecting them along and perpendicular to n^a [53]. One of the main complications when solving the

GRMHD equations numerically is preserving the constraint

$$\partial_i(\sqrt{\gamma}B^i) = 0. \quad (3.18)$$

Analytically, initial data evolved using the dynamical Maxwell equations are guaranteed to preserve the constraint. However, numerical errors generate constraint violations that need to be controlled. We opt to use the divergence cleaning method [55] where an additional field Φ is evolved in order to propagate constraint violations out of the domain. The augmented system can still be written in flux-balanced form, where the conserved variables are

$$\sqrt{\gamma}U = \sqrt{\gamma} \begin{pmatrix} D \\ S_i \\ \tau \\ B^i \\ \Phi \end{pmatrix} = \begin{pmatrix} \tilde{D} \\ \tilde{S}_i \\ \tilde{\tau} \\ \tilde{B}^i \\ \tilde{\Phi} \end{pmatrix} = \sqrt{\gamma} \begin{pmatrix} \rho W \\ (\rho h)^* W^2 v_i - \alpha b^0 b_i \\ (\rho h)^* W^2 - p^* - (\alpha b^0)^2 - \rho W \\ B^i \\ \Phi \end{pmatrix}, \quad (3.19)$$

with corresponding fluxes

$$\sqrt{\gamma}F^i = \begin{pmatrix} \tilde{D}v_{\text{tr}}^i \\ \tilde{S}_j v_{\text{tr}}^i + \alpha\sqrt{\gamma}p^*\delta_j^i - \alpha b_j \tilde{B}^i/W \\ \tilde{\tau}v_{\text{tr}}^i + \alpha\sqrt{\gamma}p^*v^i - \alpha^2 b^0 \tilde{B}^i/W \\ \tilde{B}^j v_{\text{tr}}^i - \alpha v^j \tilde{B}^i + \alpha\gamma^{ij}\tilde{\Phi} \\ \alpha\tilde{B}^i - \tilde{\Phi}\beta^i \end{pmatrix}, \quad (3.20)$$

and corresponding sources

$$\sqrt{\gamma}S = \begin{pmatrix} 0 \\ (\alpha/2)\tilde{S}^{kl}\partial_i\gamma_{kl} + \tilde{S}_k\partial_i\beta^k - \tilde{E}\partial_i\alpha \\ \alpha\tilde{S}^{kl}K_{kl} - \tilde{S}^k\partial_k\alpha \\ -\tilde{B}^j\partial_j\beta^i + \Phi\partial_k(\alpha\sqrt{\gamma}\gamma^{ik}) \\ \alpha\tilde{B}^k\partial_k\ln\alpha - \alpha K\tilde{\Phi} - \alpha\kappa\tilde{\Phi} \end{pmatrix}. \quad (3.21)$$

In deriving these equations, we have chosen to set the speed of the divergence cleaning mode to the speed of light. The transport velocity is defined as $v_{\text{tr}}^i = \alpha v^i - \beta^i$ and the generalized energy \tilde{E} and source \tilde{S}^{ij} are given by

$$\tilde{E} = \tilde{\tau} + \tilde{D}, \quad (3.22)$$

$$\tilde{S}^{ij} = \sqrt{\gamma} [(\rho h)^* W^2 v^i v^j + p^* \gamma^{ij} - \gamma^{ik} \gamma^{jl} b_k b_l]. \quad (3.23)$$

The 3+1 GRMHD divergence cleaning evolution equations analytically preserve the constraint (3.18), while numerically constraint-violating modes will be damped at a rate κ . We typically choose $\kappa \in [0, 1]$. We note that the divergence cleaning method was recently shown to be strongly hyperbolic [56], a necessary condition for a well-posed evolution problem. The primitive variables of the GRMHD system are ρ , v_i , B^i , Φ , and the specific internal energy ϵ .

Approximate Riemann solvers use the system characteristic speeds, which in the GRMHD case require solving a nontrivial quartic equation for the fast and slow modes. Instead, we use the approximation [57]:

$$\lambda_1 = -\alpha - \beta_n, \quad (3.24)$$

$$\lambda_2 = \alpha \Lambda^- - \beta_n, \quad (3.25)$$

$$\lambda_{3,4,5,6,7} = \alpha v_n - \beta_n, \quad (3.26)$$

$$\lambda_8 = \alpha \Lambda^+ - \beta_n, \quad (3.27)$$

$$\lambda_9 = \alpha - \beta_n, \quad (3.28)$$

where β_n and v_n are the shift and spatial velocity projected along the normal vector in the direction that we want to compute the characteristic speeds along, and

$$\Lambda^\pm = \frac{1}{1 - v^2 c_s^2} \left[v_n (1 - c_s^2) \pm c_s \sqrt{(1 - v^2) (1 - v^2 c_s^2 - v_n^2 (1 - c_s^2))} \right], \quad (3.29)$$

where c_s is the sound speed given by

$$c_s^2 = \frac{1}{h} \left[\left(\frac{\partial p}{\partial \rho} \right)_\epsilon + \frac{p}{\rho^2} \left(\frac{\partial p}{\partial \epsilon} \right)_\rho \right]. \quad (3.30)$$

3.3 Methods

3.3.1 Discontinuous Galerkin

The algorithm

Following [26, 29], this section describes the nodal DG method we have implemented. The algorithm is derived using the following steps (details in [26]):

- Divide the spatial domain into elements. Each element is a mapping of a reference cube with extents $[-1, 1]$ in each direction. The mapping is some time-independent function

$$\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}) \quad (3.31)$$

with Jacobian matrix

$$\mathbf{J} = \left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \right) \quad (3.32)$$

and Jacobian

$$J = \det \mathbf{J}. \quad (3.33)$$

Here the $\boldsymbol{\xi}$ -coordinates ($\boldsymbol{\xi} = (\xi, \eta, \zeta)$) are standard Cartesian-like coordinates covering the reference element.

- In each element, each component of the quantities $\sqrt{\gamma}U$, $\sqrt{\gamma}F^i$, and $\sqrt{\gamma}S$ is expanded in polynomial basis functions. We will use s and t to be grid point indices. That is, U_s is U at the grid point s . For tensor product bases we index the individual irreducible topologies by subscripting s . For example, s_1 would be the ξ -dimension in the reference element. We choose these basis functions to be a tensor product of 1d basis functions ℓ_{s_i} on the reference element, so that the expansion of a typical variable takes the form

$$U(\boldsymbol{\xi}) = \sum_s U_s(t) \ell_{s_1}(\xi) \ell_{s_2}(\eta) \ell_{s_3}(\zeta), \quad (3.34)$$

where the time-dependent coefficients $U_s(t)$ are found from (3.40) below and $U_s(0)$ is given in terms of the initial data. The 1d basis functions of degree N are simply the Lagrange interpolating polynomials corresponding to Legendre polynomials,

$$\ell_{s_i}(\xi) = \prod_{\substack{t_i=0 \\ t_i \neq s_i}}^N \frac{\xi - \xi_{t_i}}{\xi_{s_i} - \xi_{t_i}}, \quad (3.35)$$

where ξ_{s_i} are the nodes of a Legendre-Gauss-Lobatto (LGL) quadrature. These nodes may be found with standard algorithms, for example Algorithm 24 of [58]. We denote a DG scheme with basis functions of degree N by P_N . A P_N scheme is expected to converge at order $\mathcal{O}(\Delta x^{N+1})$ for smooth solutions [29], where Δx is the 1d size of the element.

- In each element, we follow the standard DG procedure of integrating (3.2) multiplied by a basis function over the proper volume $\sqrt{\gamma}d^3x$ of the element, where d^3x is the coordinate volume element $dx dy dz$,

$$\int [\partial_t(\sqrt{\gamma}U) + \partial_i(\sqrt{\gamma}F^i) - \sqrt{\gamma}S] \phi_s(\mathbf{x}) d^3x = 0. \quad (3.36)$$

Use integration by parts (Gauss's Theorem) to convert the divergence term to a surface integral:

$$\begin{aligned} \int \partial_i(\sqrt{\gamma}F^i) \phi_s(\mathbf{x}) d^3x &= \int \partial_i(\sqrt{\gamma}F^i \phi_s(\mathbf{x})) d^3x - \int \sqrt{\gamma}F^i \partial_i \phi_s(\mathbf{x}) d^3x \\ &= \oint F^i n_i \phi_s d^2\Sigma - \int \sqrt{\gamma}F^i \partial_i \phi_s(\mathbf{x}) d^3x. \end{aligned} \quad (3.37)$$

Here $d^2\Sigma$ is the proper surface element of the cell and in this section we use n_i as the unit outward normal to the element.

With a formulation like (3.37) in each element, there is no connection between the elements. The heart of the DG method is to replace $F^i n_i$ in the surface term by a numerical flux G , a function of the state vector on *both* sides of the

interface:

$$\int \partial_i (\sqrt{\gamma} F^i) \phi_s(\mathbf{x}) d^3x \rightarrow \oint G \phi_s d^2\Sigma - \int \sqrt{\gamma} F^i \partial_i \phi_s(\mathbf{x}) d^3x. \quad (3.38)$$

We note that replacing only F^i with a numerical flux term as is done in [26] is incorrect when the metric is dynamical. Next we undo the integration by parts on the right-hand side of (3.38):

$$\int \partial_i (\sqrt{\gamma} F^i) \phi_s(\mathbf{x}) d^3x \rightarrow \oint (G - F^i n_i) \phi_s d^2\Sigma + \int \partial_i (\sqrt{\gamma} F^i) \phi_s(\mathbf{x}) d^3x. \quad (3.39)$$

- Evaluate the integrals by using the expansion in basis functions and LGL quadrature, mapping to the reference element with (3.31). The final result is Eq. 3.17 of [26]:

$$\begin{aligned} & \frac{d(\sqrt{\gamma}U)_s}{dt} + \left[\frac{\partial \xi}{\partial x^i} \Big|_s \sum_{t_1} D_{s_1 t_1}^\xi (\sqrt{\gamma} F^i)_{t_1 s_2 s_3} + \frac{\partial \eta}{\partial x^i} \Big|_s \sum_{t_2} D_{s_2 t_2}^\eta (\sqrt{\gamma} F^i)_{s_1 t_2 s_3} \right. \\ & \quad \left. + \frac{\partial \zeta}{\partial x^i} \Big|_s \sum_{t_3} D_{s_3 t_3}^\zeta (\sqrt{\gamma} F^i)_{s_1 s_2 t_3} \right] - (\sqrt{\gamma}S)_s \\ &= -\frac{1}{w_N} F_{s_1 s_2 N} \frac{\sqrt{{}^{(2)}\gamma_{s_1 s_2}}}{J_{s_1 s_2 N}} \delta_{s_3 N} + \frac{1}{w_0} F_{s_1 s_2 0} \frac{\sqrt{{}^{(2)}\gamma_{s_1 s_2}}}{J_{s_1 s_2 0}} \delta_{s_3 0} - \frac{1}{w_N} F_{N s_2 s_3} \frac{\sqrt{{}^{(2)}\gamma_{s_2 s_3}}}{J_{N s_2 s_3}} \delta_{s_1 N} \\ & \quad + \frac{1}{w_0} F_{0 s_2 s_3} \frac{\sqrt{{}^{(2)}\gamma_{s_2 s_3}}}{J_{0 s_2 s_3}} \delta_{s_1 0} - \frac{1}{w_N} F_{s_1 N s_3} \frac{\sqrt{{}^{(2)}\gamma_{s_1 s_3}}}{J_{s_1 N s_3}} \delta_{s_2 N} + \frac{1}{w_0} F_{s_1 0 s_3} \frac{\sqrt{{}^{(2)}\gamma_{s_1 s_3}}}{J_{s_1 0 s_3}} \delta_{s_2 0}. \end{aligned} \quad (3.40)$$

Here $D_{s_i t_i}^\xi$ is the differentiation matrix

$$D_{s_i t_i}^\xi = \frac{\partial \ell_{t_i}(\xi)}{\partial \xi} \Big|_{s_i} \quad (3.41)$$

for ξ , and similarly for the η - and ζ -coordinates. The quantity F is the normal component of the flux difference, $F = (G - F^i n_i)$. The quantity ${}^{(2)}\gamma$ is the determinant of the 2-dimensional metric induced on the surface by γ_{ij} , and w_0 and w_N are the weights of the Gauss-Lobatto quadrature at the endpoints of the interval.

The semi-discrete system (3.40) is integrated in time using a method-of-lines strategy. Section 3.3.4 summarizes the time steppers we use.

Note that in the derivation of (3.40), each product of expansions is evaluated using a single expansion with coefficients equal to the product of the original coefficients. This replacement leads to an aliasing error: contributions from the high order polynomials are aliased back onto the basis. While this does not affect the precision of the scheme, it can lead to an aliasing-driven instability, which may need to be dealt with by filtering [29].

Implementation in C++

Given the number of matrix multiplications present in the DG algorithm (3.40), it is important to make sure these are computed as efficiently as possible, while ensuring that the flux and source terms are still able to be computed efficiently. In order to obtain good performance both during differentiation and during flux and source term computations, the values at grid points are stored contiguously in xyz -order (x varies fastest in memory). The derivative matrices are precomputed at the beginning of the simulation. Naively the matrices would be applied to each component of U and F^i as a matrix-vector operation. However, since derivatives of all variables are typically needed (or at least all fluxes), the matrix-vector multiplication can be done more efficiently as a matrix-matrix multiplication. We do this by storing the quantities to differentiate in contiguous memory in $xyzn$ -order, where n labels the individual variable components. Specifically, for a 3-dimensional rank-1 tensor T^i we store contiguously the values of T^x over the grid, followed by the values of T^y , followed by the values of T^z . When computing 3d partial derivatives, we first apply the differentiation matrix in x , then transpose the data so that y varies fastest, apply the differentiation matrix in y , transpose the data so that z varies fastest, apply the differentiation matrix in z , and finally transpose

back to x varying fastest. The matrix-matrix multiplication consists of two small matrices, size $(N + 1) \times (N + 1)$ for the derivative matrix ($N \sim \mathcal{O}(2)$) and size $(N + 1) \times (\text{Number of Variable Components})$ for the variables being differentiated. For the GRMHD system there are nine components, and when the first-order generalized harmonic equations [59] for GR are also evolved this increases to 59 components. We use the LIBXSMM library [60–63] to obtain very efficient small matrix-matrix multiplications. Finally, after the derivative matrix is applied we have the logical derivatives, which need to be multiplied by the Jacobian matrix. This is done for each variable and is vectorized using the Blaze library [64, 65].

The motivation for storing the variables, fluxes, sources, etc. in large contiguous buffers is threefold. First, it reduces the number of memory allocations necessary, which, even with fast allocators like jemalloc [66] and tcmalloc [67], are expensive in multithreaded environments. Second, making fewer large allocations reduces memory fragmentation compared to making many small allocations. Finally, storing the variables contiguously allows for efficient differentiation, interpolation, and projection, since all these operations can be cast as dense matrix-matrix multiplications instead of matrix-vector operations.

Each evolution system in SpECTRE implements a function that computes the fluxes F^i and another that computes the source terms S . Generic code then slices the contributions of the fluxes and variables needed for computing the numerical flux to the boundaries of the elements. We describe numerical fluxes we have implemented in Section 3.3.5. Additionally, each evolution system that has both primitive and conservative variables must implement functions to compute the one set of variables from the other.

We summarize our task-based algorithm implemented in SpECTRE in Algorithm 3.1. Limiting is discussed in Section 3.3.3, and is not needed for problems

```

1: Member Function StartTimeStep( $t^n$ )
2:   Compute  $F^i$ ,  $S$  from  $U^n$ 
3:    $F_{\text{boundary}}^i \leftarrow$  compute  $F^i$  on element boundaries
4:    $G_{\text{local}}^n \leftarrow$  compute local contribution to  $G$ 
5:   for all neighbors  $\Omega_j$  do
6:      $\Omega_j$ .EndTimeStep( $G_{\text{local}}^n$ ,  $\Omega_{\text{self}}$ )
7:   end for
8:    $dU^n/dt \leftarrow S - \partial_i F^i$ 
9:
10: Member Function EndTimeStep( $G_{\text{nhbr}}^n$ ,  $\Omega_{\text{nhbr}}$ )
11: if received all  $G_{\text{nhbr}}^n$  and  $G_{\text{local}}^n$  is computed then
12:   for all neighbors do
13:      $G_{\text{boundary}} \leftarrow$  combine  $G_{\text{local}}$  and  $G_{\text{nhbr}}$ 
14:      $dU^n/dt \leftarrow dU^n/dt + (G_{\text{boundary}} - F_{\text{boundary}}^i n_i)$ 
15:   end for
16:    $U^{n+1} \leftarrow$  integrate in time one step
17:   if using limiter then
18:      $L_{\text{local}}^{n+1} \leftarrow$  compute local limiter contribution
19:     for all neighbors  $\Omega_j$  do
20:        $\Omega_j$ .ApplyLimiter( $L_{\text{local}}^{n+1}$ ,  $\Omega_{\text{self}}$ )
21:     end for
22:   else
23:      $\Omega_{\text{self}}$ .StartTimeStep( $t^{n+1}$ )
24:   end if
25: else
26:   Store  $G_{\text{nhbr}}^n$ 
27: end if
28:
29: Member Function ApplyLimiter( $L_{\text{nhbr}}^{n+1}$ ,  $\Omega_{\text{nhbr}}$ )
30: if received all  $L_{\text{nhbr}}^{n+1}$  at time  $t^{n+1}$  then
31:    $U^{n+1} \leftarrow$  limit  $U^{n+1}$  using  $L_{\text{local}}^{n+1}$  and  $L_{\text{nhbr}}^{n+1}$ 
32:    $\Omega_{\text{self}}$ .StartTimeStep( $t^{n+1}$ )
33: else
34:   Store  $L_{\text{nhbr}}^{n+1}$ 
35: end if
36:

```

ALGORITHM 3.1: The task-based algorithm implemented in SpECTRE. The superscript n is used to denote the time level. Each function is an individual task in the algorithm and is evaluated on the element by Charm++ once the data has been received. While one element is passively waiting for data, Charm++ evaluates functions on elements whose data is available.

that do not form shocks, and so we skip limiting for smooth problems. We distribute the elements in such a manner that in a typical simulation each core has between several and several thousand elements assigned to it. Each element then performs Algorithm 3.1 until the end of the simulation. Having many elements per core means that each core typically has a task to perform (e.g. EndTimeStep) on one element while other elements are waiting on data from their neighbors. Having several tasks to perform per core while waiting on data is what ultimately allows task-based parallelism to mask the communication.

3.3.2 Domain

General Block Decomposition

As described in Section 3.3.1, the physical domain is decomposed into several non-overlapping hexahedral elements. In SpECTRE, the initial elements resulting from the decomposition are referred to as *blocks*, which are constructed such that block boundaries lie along the entire face of each adjacent block. Each block has its own “logical” coordinate system $\boldsymbol{\xi} = (\xi, \eta, \zeta) \in [-1, 1]^3$. In a domain with multiple blocks, the logical coordinates used within each block are not necessarily aligned on shared block boundaries. We use a generalization of the corner-numbering scheme described in [58] to properly communicate information across the boundaries of adjacent blocks with differently oriented logical coordinate systems. Each block is further divided into the $2^{\ell_\xi + \ell_\eta + \ell_\zeta}$ elements on which the DG scheme is applied, where ℓ_ξ , etc. are the levels of refinement along each logical dimension. While the number of elements may span into the millions, the number of blocks needed to partition astrophysically relevant physical domains (e.g. disks, stars) into hexahedra is often less than ten. The number of unique coordinate maps needed is at most the number of blocks used; individual element maps are merely the block

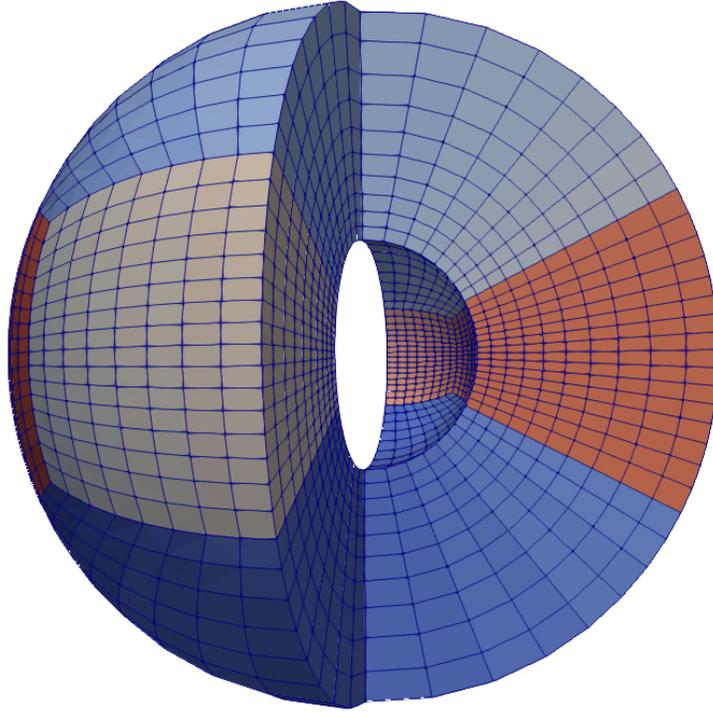


Figure 3.1: A cutaway of a cubed spherical shell domain. The six blocks in the domain are indicated by the colors. Only half of each block is shown. An angular compression map with aspect ratio $\alpha = 2$ and an exponential mapping in the radial direction have been used to distribute the points appropriately for a disk simulation.

maps restricted to a region of the reference cube. Since [27] we have implemented coordinate maps for curvilinear elements. In Figure 3.1 we show a cubed spherical shell computational domain as an example of a curvilinear domain.

Cubed Spherical Shells

The computational domains we will be using most often in our compact object simulations are based on the cubed sphere [68]. SpECTRE black hole simulations require the singularity within the hole to be excised from the domain [69, 70], and we do so with a shell partitioned using a cubed sphere decomposition, as shown in Figure 3.1. Cubed spherical shell domains are commonly used in accretion disk

simulations. The cubed sphere map partitions the sphere into six identical regions that we extend radially to form a shell from R_{in} to R_{out} . The individual blocks are oriented such that ξ and η are angular coordinates, and ζ is the radial coordinate. We are free to choose the parameterization $R(\zeta)$ of the radial coordinate; a parameterization that we frequently use is an exponential map

$$R(\zeta) = \sqrt{R_{\text{in}}^{1-\zeta} R_{\text{out}}^{1+\zeta}} \quad (3.42)$$

that concentrates grid points closer to the center.

We define the cubed sphere map in terms of the variables Ξ and H (x/a and y/a in C1 of [68]), which we define as

$$\Xi(\xi) = \tan(\xi\pi/4), \quad (3.43)$$

$$H(\eta) = \tan(\eta\pi/4). \quad (3.44)$$

For the block lying in the $+z$ direction, the mapping is

$$\mathbf{x}(\boldsymbol{\xi}) = \frac{R(\zeta)}{\sqrt{1 + \Xi^2 + H^2}} \begin{bmatrix} \Xi \\ H \\ 1 \end{bmatrix}. \quad (3.45)$$

For the other blocks, this map is applied followed by a rotation about the coordinate center of the shell.

Redistribution of Grid Points

We subsequently adjust the polar angles using a mapping $\tan \theta' = \alpha \tan \theta$ that moves grid points near the poles towards the equator. This mapping is determined by a single parameter, α , which we call the *aspect ratio* of the map, because under this map a square centered at the origin would be mapped to a rectangle with an

aspect ratio α . In Cartesian coordinates, the mapping is:

$$\mathbf{x}'(x, y, z) = \sqrt{\frac{x^2 + y^2 + z^2}{x^2 + y^2 + \alpha^{-2}z^2}} \begin{bmatrix} x \\ y \\ \alpha^{-1}z \end{bmatrix}. \quad (3.46)$$

The resulting domain is shown in Figure 3.1.

3.3.3 Limiting

Near shocks or surfaces in the fluid, the DG solution may exhibit spurious oscillations (i.e., Gibbs phenomenon) and overshoots. These oscillations can lead to a non-physical fluid state (e.g., negative densities) at individual grid points and prevent stable evolution of the system. To maintain a stable scheme, we limit the solution: we identify elements where the solution contains spurious oscillations (we label these elements as “troubled cells”) and we modify the solution on these elements to reduce the amount of oscillation.

In this work we consider limiters that preserve the order of the DG solution while maintaining a compact (nearest-neighbor) stencil. The compact stencil greatly simplifies communication patterns, but, in order to provide the limiter with sufficient information to preserve the order of the scheme, it becomes necessary to send larger amounts of data from each element for each limiting step. We specifically consider the hierarchical limiter of Krivodonova [71], and two different formulations of weighted essentially non-oscillatory (WENO) limiters for DG: the simple WENO limiter of [72] and the Hermite WENO (HWENO) limiter of [73]. Below we summarize the action of these three limiters.

The Krivodonova limiter works by limiting the coefficients of the solution’s modal representation, starting with the highest coefficient then decreasing in order until no more limiting is necessary. This procedure is repeated independently for

each variable component U being limited. Although it is only given in one or two dimensions, the limiting algorithm is straightforwardly generalized to our 3D application. We expand U over a basis of Legendre polynomials P_i ,

$$U^{l,m,n} = \sum_{i,j,k=0,0,0}^{N,N,N} c_{i,j,k}^{l,m,n} P_i(\xi) P_j(\eta) P_k(\zeta), \quad (3.47)$$

where the $c_{i,j,k}^{l,m,n}$ are the modal coefficients, with the superscript $\{l, m, n\}$ representing the element indexed by l, m, n , and the upper bound N is the number of collocation points minus one in each of the ξ, η, ζ directions.

Each coefficient is limited by comparison with the coefficients of U in neighboring elements. The new value $\tilde{c}_{i,j,k}^{l,m,n}$ of $c_{i,j,k}^{l,m,n}$ is computed according to

$$\begin{aligned} \tilde{c}_{i,j,k}^{l,m,n} = \min\text{mod} & \left(c_{i,j,k}^{l,m,n}, \alpha_i \left(c_{i-1,j,k}^{l+1,m,n} - c_{i-1,j,k}^{l,m,n} \right), \alpha_i \left(c_{i-1,j,k}^{l,m,n} - c_{i-1,j,k}^{l-1,m,n} \right), \right. \\ & \alpha_j \left(c_{i,j-1,k}^{l,m+1,n} - c_{i,j-1,k}^{l,m,n} \right), \alpha_j \left(c_{i,j-1,k}^{l,m,n} - c_{i,j-1,k}^{l,m-1,n} \right), \\ & \left. \alpha_k \left(c_{i,j,k-1}^{l,m,n+1} - c_{i,j,k-1}^{l,m,n} \right), \alpha_k \left(c_{i,j,k-1}^{l,m,n} - c_{i,j,k-1}^{l,m,n-1} \right) \right), \quad (3.48) \end{aligned}$$

where $\min\text{mod}$ is the $\min\text{mod}$ function defined as

$$\min\text{mod}(a, b, \dots) = \begin{cases} \text{sgn}(a) \min(|a|, |b|, \dots), & \text{if } \text{sgn}(a) = \text{sgn}(b) = \text{sgn}(\dots) \\ 0, & \text{otherwise,} \end{cases} \quad (3.49)$$

and the $\alpha_i, \alpha_j, \alpha_k$ set the strength of the limiter. In all cases shown in this paper, we set $\alpha_i = 1$, at the least dissipative end of the range for these parameters³.

The algorithm for limiting from highest to lowest modal coefficient is as follows. We first compute $\tilde{c}_{N,N,N}$ (we drop the element superscripts here). If this is equal to $c_{N,N,N}$, no limiting is done. Otherwise, we update $c_{N,N,N} = \tilde{c}_{N,N,N}$, and compute the trio of coefficients $\tilde{c}_{N,N,N-1}, \tilde{c}_{N,N-1,N}, \tilde{c}_{N-1,N,N}$. If *all* of these are unchanged,

³Whereas Krivodonova [71] changes normalization convention for the Legendre polynomials in going from one to two dimensions, our convention matches their 1D convention in all cases, so that the range of the α_i parameters are given by Eq. (14) in the reference.

the limiting stops. Otherwise, we update each coefficient and proceed to limiting the trios given by index permutations of $c_{N,N,N-2}$, ..., $c_{N,N,0}$, $c_{N,N-1,N-1}$, in turn, up to the three index permutations of $c_{1,0,0}$. Finally, the limited modal coefficients are used to recover the limited nodal values of the function U .

For the two WENO limiters, we use a troubled-cell indicator to identify whether limiting is needed. We use a TVB minmod limiter [31, 74, 75] as troubled cell indicator; if at least one quantity U to be limited is flagged for slope reduction by the minmod limiter, then the element is labeled as a troubled cell, and every quantity U is reconstructed using the WENO procedure.

We summarize first the simple WENO limiter [72]. For its reconstruction, this limiter uses several different estimates for U on the troubled element labeled by k . The first of these is the local data U^k . Each neighbor n of k also provides a “modified” solution estimate U^{k_n} ; in the case of the simple WENO limiter, this estimate is simply obtained by evaluating the neighbor’s solution U^n on the grid points of the element k . We follow the standard WENO algorithm of reconstructing the solution from a weighted sum of these estimates,

$$U_{\text{new}}^k = \omega_k U^k + \sum_n \omega_n U^{k_n}, \quad (3.50)$$

where the ω_i are the weights associated with each solution estimate, and satisfy the normalization $\sum_i \omega_i = 1$.

The weights are obtained by first computing an oscillation indicator (also called a smoothness indicator) σ_i for each $U^i = \{U^k, U^{k_n}\}$, which measures the amount of oscillation in the data. We use an indicator based on Eq. (23) of [76], but adapted for use on square or cubical grids,

$$\sigma_i = \sum_{\alpha=0}^N \sum_{\beta=0}^N \sum_{\substack{\gamma=0 \\ \alpha+\beta+\gamma>0}}^N \int \left(\frac{\partial^{\alpha+\beta+\gamma}}{\partial \xi^\alpha \partial \eta^\beta \partial \zeta^\gamma} U^i \right)^2 d\xi d\eta d\zeta. \quad (3.51)$$

Here the restriction on the sum avoids the term that has no derivatives of U^i . From

the oscillation indicators, we compute the non-linear weights

$$\bar{\omega}_i = \frac{\gamma_i}{(\epsilon + \sigma_i)^2}. \quad (3.52)$$

Here the γ_i are the linear weights that give the relative weight of the local and neighbor contributions before accounting for oscillation in the data, and ϵ is a small number to avoid the denominator vanishing. We use standard values from the literature for both — we take $\gamma_{k_n} = 0.001$ for the neighbor contributions (then $\gamma_k = 0.994$ for an element with 6 neighbors; in general γ_k is set by the requirement that all the γ_i sum to unity), and $\epsilon = 10^{-6}$. Finally, the normalized non-linear weights that go into the WENO reconstruction are given by

$$\omega_i = \frac{\bar{\omega}_i}{\sum_i \bar{\omega}_i}. \quad (3.53)$$

Our implementation of the HWENO limiter [73] follows similar steps. Note that we again use the TVB minmod limiter as troubled-cell indicator, whereas the reference uses the troubled-cell indicator of [77]. The HWENO modified solution estimates from the neighboring elements are computed as a least-squared fit to U across several elements. This broader fitting reduces oscillations as compared to the polynomial extrapolation used in the simple WENO estimates, and this improves robustness near shocks. From the estimates, the WENO reconstruction is performed as for simple WENO.

Because computing the characteristic variables of the GRMHD system is complicated, we apply the limiter to the evolved (i.e., conserved) variables. However, we do not limit the divergence-cleaning field Φ , as it is not expected to form any shocks. The limiter is applied at the end of each time step when using an Adams-Bashforth method, and at the end of each substep when using a multi-step Runge-Kutta method.

3.3.4 Time Stepping

SpECTRE supports time integration using explicit multistep and substep integrators. The results presented here were obtained using either a self-starting Adams-Bashforth method or a strong stability-preserving third-order Runge-Kutta method [29]. SpECTRE additionally supports local time-stepping when using Adams-Bashforth schemes [78], but that feature was not used for any of these problems. The maximum admissible time step size for a P_N scheme is

$$\Delta t \leq \frac{c}{d(2N + 1)} \frac{\Delta x}{\lambda_{\max}}, \quad (3.54)$$

where c is a constant dependent on the time stepper used, d is the number of spatial dimensions, Δx is the minimum size of the element, and λ_{\max} is the maximum characteristic speed in the element of all the variables.

3.3.5 Numerical Fluxes

One of the key ingredients in conservative numerical schemes is the approximate solution to the Riemann problem on the interface. We use general implementations of the Rusanov numerical flux [79] (also known as the local Lax-Friedrichs flux), and the numerical flux of Harten, Lax, and van Leer (HLL) [80, 81]. The Rusanov flux is given by

$$G^{\text{Rusanov}} = \frac{1}{2} (F^{k,+} n_k^+ + F^{k,-} n_k^-) - \frac{C}{2} (u^+ - u^-),$$

where $C = \max(|\lambda_i(u^+)|, |\lambda_i(u^-)|)$, and $\lambda_i(u)$ is the characteristic speed of the field u . Quantities superscripted with a plus sign are on the exterior side of the mortar, while quantities superscripted with a minus sign are on the interior side. In this section n_k is the outward pointing unit normal to the element.

The HLL flux is given by

$$G^{\text{HLL}} = \frac{c_{s,\min} F^{k,+} n_k^+ + c_{s,\max} F^{k,-} n_k^-}{c_{s,\max} - c_{s,\min}} - \frac{c_{s,\max} c_{s,\min}}{c_{s,\max} - c_{s,\min}} (u^+ - u^-), \quad (3.55)$$

where $c_{s,\min}$ and $c_{s,\max}$ are estimates for the fastest left- and right-moving signal speeds, respectively. We compute the approximate signal speeds pointwise using the scheme presented in reference [82]. Specifically,

$$\begin{aligned} c_{s,\min} &= \min(\lambda_i(u^+), \lambda_i(u^-), 0), \\ c_{s,\max} &= \max(\lambda_i(u^+), \lambda_i(u^-), 0). \end{aligned} \quad (3.56)$$

Other numerical fluxes will be considered in future work.

3.3.6 Primitive Recovery

One of the most difficult and expensive aspects of evolving the GRMHD equations is recovering the primitive variables from the conserved variables. Several different primitive recovery schemes are compared in [83]. Of the ones compared in [83] we currently use the Newman-Hamlin scheme [84] and the scheme of Palenzuela *et al.* [85].

3.3.7 Variable Fixing

During the evolution the conserved and primitive variables can reach states that are non-physical or enter regimes where the evolution is no longer stable (e.g. zero density). When limiting the solution does not remove these unphysical or bad values a pointwise fixing procedure is used where at any grid points where the chosen conditions are not satisfied the variables are adjusted. The fixing procedures are generally not conservative and are used only as a fallback to ensure a stable evolution. In SpECTRE we currently use two fixing algorithms: the first applies

an “atmosphere” in low-density regions, while the second adjusts the conserved variables in an attempt to guarantee primitive recovery.

Our “atmosphere” treatment is similar to that of [86]. We define values ρ_{atm} and ρ_{cutoff} where $\rho_{\text{atm}} \leq \rho_{\text{cutoff}}$. For any point where $\rho < \rho_{\text{cutoff}}$ we set

$$\rho = \rho_{\text{atm}}, \quad (3.57)$$

$$v^i = 0, \quad (3.58)$$

$$W = 1. \quad (3.59)$$

After the primitive variables are set to the atmosphere we recompute the conserved variables from the primitive ones.

Our fixing of the conserved variables is based on that of references [87, 88]. We define D_{min} and D_{cutoff} and adjust \tilde{D} if $D < D_{\text{cutoff}}$. Specifically, we set $\tilde{D} = \sqrt{\gamma} D_{\text{min}}$. We adjust $\tilde{\tau}$ such that $\tilde{B}^2 \leq 2\sqrt{\gamma}(1 - \epsilon_B)\tilde{\tau}$, where ϵ_B is a small number typically set to 10^{-12} .

Finally, we adjust \tilde{S}_i such that $\tilde{S}^2 \leq \tilde{S}_{\text{max}}^2$, where \tilde{S}_{max}^2 is defined below. We define variables

$$\hat{\tau} = \frac{\tilde{\tau}}{\tilde{D}}, \quad (3.60)$$

$$\hat{B}^2 = \frac{\tilde{B}^2}{\sqrt{\gamma}\tilde{D}}, \quad (3.61)$$

$$\hat{\mu} = \begin{cases} \frac{\tilde{S}_i \tilde{B}^i}{\sqrt{\tilde{B}^2 \tilde{S}^2}}, & \tilde{B}^2 > \tilde{D} \times 10^{-16} \text{ and } \tilde{S}^2 > \tilde{D}^2 \times 10^{-16} \\ 0, & \text{otherwise} \end{cases} \quad (3.62)$$

The Lorentz factor is bounded by

$$\max(1, 1 + \hat{\tau} - \hat{B}^2) \leq W \leq 1 + \hat{\tau}, \quad (3.63)$$

and is determined by finding the root of

$$g(W) = \left(W + \hat{B}^2 - \hat{\tau} - 1 \right) \left[W^2 + \hat{B}^2 \hat{\mu}^2 (\hat{B}^2 + 2W) \right] - \frac{\hat{B}^2}{2} \left[1 + \hat{\mu}^2 \left(W^2 + 2W \hat{B}^2 + \hat{B}^4 - 1 \right) \right]. \quad (3.64)$$

Using the Lorentz factor W obtained by solving (3.64) we defined \tilde{S}_{\max} as

$$\tilde{S}_{\max} = \min \left(1, \sqrt{\frac{(1 - \epsilon_S) (W + \hat{B})^2 (W^2 - 1) \tilde{D}}{(\tilde{S}^2 + \tilde{\tau}^2 \times 10^{-16}) [W^2 + \hat{\mu} \hat{B}^2 (\hat{B}^2 + 2W)]}} \right), \quad (3.65)$$

where ϵ_S is a small number typically set to 10^{-12} . We apply the check on the conservatives after each time step before a primitive recovery.

3.3.8 Parallelism

We use the Charm++ parallel runtime system for parallelization [89]. Charm++ allows us to define C++ objects that are distributed across the supercomputer. In SpECTRE we refer to these as *distributed objects*, while Charm++ calls them *chares*. There are four different types of distributed objects provided by Charm++: *singletons* (one per execution), dynamically sized *arrays*, one-per-core arrays called *groups*, and one-per-node arrays called *nodegroups*. There can be thousands of these distributed objects per core. Each distributed object has member functions that perform a computation like evaluating the time derivative, computing the solution at the next time, and interpolating data for observation. We call a collection of distributed objects of the same type and belonging to the same array a *parallel component* (singletons are included in this definition as arrays with a single element). Thus, we have singleton, array, group, and nodegroup parallel components and each simulation can have any number of different parallel components.

We have built a layer on top of Charm++ that handles various tasks and eliminates bugs we frequently encountered while developing the closed-source version of SpECTRE [27]. As part of developing this layer we made user interactions with any parallelization infrastructure as minimal as possible. As a result, most people developing code in SpECTRE will only be writing free functions or small classes. We have found that this lowers the barrier for new users to contribute physics modules to SpECTRE.

The overall communication pattern has not changed much since [27]. We have stopped using a separate parallel component for the mortars between elements because we found removing them reduced communication overhead and simplified the bookkeeping necessary for implementing adaptive mesh refinement.

3.4 Results

3.4.1 Benchmark tests

Alfvén Wave

A finite amplitude circularly polarized Alfvén wave was first proposed as a test for the GRMHD system in [90]. The magnetic field \mathbf{B} in the Alfvén wave solution is the sum of a background static magnetic field \mathbf{B}_0 and a transverse time-dependent magnetic field \mathbf{B}_1 . We define the auxiliary magnetic velocities v_{B_0} and v_{B_1} as

$$v_{B_0}^2 = \frac{(B_0)^2}{\rho h + B^2}, \quad v_{B_1}^2 = \frac{(B_1)^2}{\rho h + B^2}. \quad (3.66)$$

The Alfvén speed and fluid speed are then given by:

$$v_A^2 = \frac{v_{B_0}^2}{\frac{1}{2} + \sqrt{\frac{1}{4} - v_{B_0}^2 v_{B_1}^2}}, \quad (3.67)$$

$$v_f^2 = \frac{v_{B_1}^2}{\frac{1}{2} + \sqrt{\frac{1}{4} - v_{B_0}^2 v_{B_1}^2}}. \quad (3.68)$$

The ideal fluid equation of state is used:

$$p = \rho \epsilon (\gamma - 1), \quad (3.69)$$

where γ is the adiabatic index.

The circularly polarized wave is best described in a basis aligned with the initial magnetic fields at the origin. To this end we define the unit vectors $\hat{\mathbf{b}}_0$, $\hat{\mathbf{b}}_1$, and $\hat{\mathbf{e}}$ as

$$\hat{\mathbf{b}}_0 = \frac{\mathbf{B}_0}{B_0} \quad (3.70)$$

$$\hat{\mathbf{b}}_1 = \frac{\mathbf{B}_1}{B_1} \Big|_{\mathbf{x}=\mathbf{0}, t=0} \quad (3.71)$$

$$\hat{\mathbf{e}} = \hat{\mathbf{b}}_0 \times \hat{\mathbf{b}}_1. \quad (3.72)$$

We test our numerical evolution of the circularly polarized Alfvén wave by comparing with the analytic solution, given by

$$\rho(\mathbf{x}, t) = \rho, \quad (3.73)$$

$$\mathbf{v}(\mathbf{x}, t) = -v_f \left[\cos(\delta\phi) \hat{\mathbf{b}}_1 + \sin(\delta\phi) \hat{\mathbf{e}} \right], \quad (3.74)$$

$$p(\mathbf{x}, t) = p, \quad (3.75)$$

$$\epsilon(\mathbf{x}, t) = \frac{p}{(\gamma - 1)\rho}, \quad (3.76)$$

$$\mathbf{B}(\mathbf{x}, t) = \mathbf{B}_0 + B_1 \left[\cos(\delta\phi) \hat{\mathbf{b}}_1 + \sin(\delta\phi) \hat{\mathbf{e}} \right], \quad (3.77)$$

where $\delta\phi$ is the phase, which evolves as

$$\delta\phi(\mathbf{x}, t) = k \left(\mathbf{x} \cdot \hat{\mathbf{b}}_0 - v_A t \right), \quad (3.78)$$

where k is the wave vector.

To test the convergence order of our DG implementation on smooth solutions, we measure the L_1 error in v_z $L_1(\mathcal{E}(v_z))$ after one period on a regular Cartesian domain of size $[0, 2\pi]^3$. We use the Rusanov flux, a third-order Adams-Bashforth time stepper with $\Delta t = 2^{-\ell}/500$, and impose analytic boundary conditions. We use the initial conditions

$$k = \sqrt{3}, \tag{3.79}$$

$$\rho = 1.0, \tag{3.80}$$

$$P = 1.0, \tag{3.81}$$

$$\gamma = 1.6, \tag{3.82}$$

$$\mathbf{B}_0 = [1, 1, 1]^T, \tag{3.83}$$

$$\mathbf{B}_1(\mathbf{x} = \mathbf{0}, t = 0) = \left[\sqrt{2}, \frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}} \right]^T. \tag{3.84}$$

In Figure 3.2 we plot $L_1(\mathcal{E}(v_z))$ as a function of the number of elements in each direction, N_x , for different P_N schemes. In each case we observe the expected convergence until we reach the error floor at $\sim 10^{-13}$. In Figure 3.2 we plot the expected convergence rates as dashed lines next to the data points as a comparison. Figure 3.3 shows that we observe exponential convergence as the order of the scheme is increased. In Table 3.1 we provide the local convergence order and $L_1(\mathcal{E}(v_z))$ for the different cases we studied. The expected convergence order is observed for the different schemes.

Shock tests

To test SpECTRE's ability to handle strong shocks we study the 1d tests of Komisarov [91]. Constant initial data is specified on both sides of a discontinuous interface located at x_d . We focus on the slow shock test with initial conditions

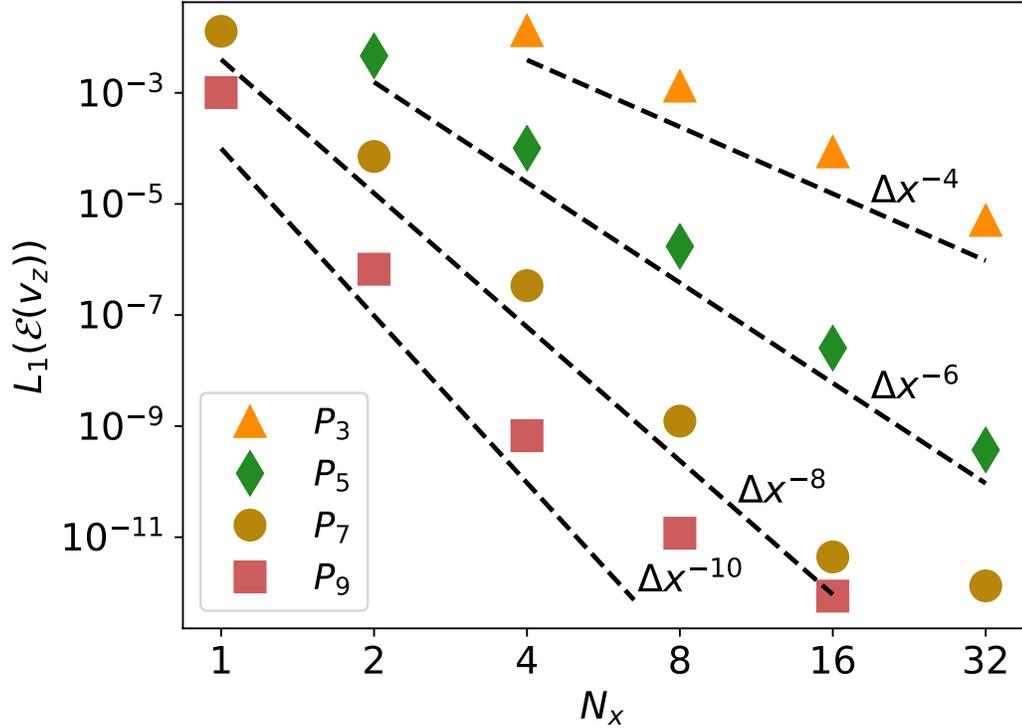


Figure 3.2: Convergence plot of the Alfvén wave at time 7.0 increasing the number of elements for different P_N schemes. The number of elements in each dimension is N_x . The dashed lines show the expected convergence rates for the different order schemes.

given in Table 3.2. We use an ideal fluid equation of state, (3.69), with $\gamma = 4/3$. We perform evolutions using the Rusanov flux, a third-order Adams-Bashforth time stepper, and the simple WENO, HWENO and Krivodonova limiters. We perform the tests in 3d with the shock propagating along the x -axis and with periodic boundary conditions in the y - and z -direction. The extent of the domain is $[-0.5, 2.06] \times [0, 1] \times [0, 1]$ with $N_x = 256$ elements in the x -direction, and one element in the y - and z -direction.

In Figure 3.4 we show the evolution of the slow shock problem at times $t = \{0, 0.48, 0.96, 1.44, 1.92\}$ compared with the analytic solution (solid lines) at those times for the simple WENO, HWENO, and Krivodonova limiters using P_1 and P_2 schemes. All limiters are able to capture the shock speed correctly and re-

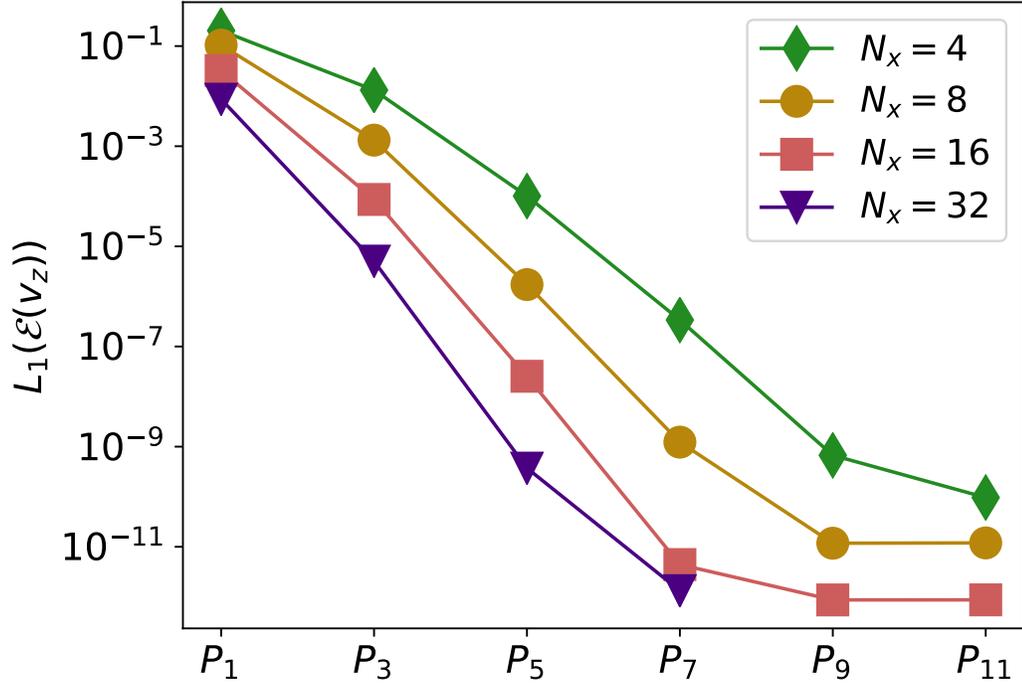


Figure 3.3: Convergence plot of the Alfvén wave at time 7.0 for increasing P_N schemes for different number of elements in each direction, N_x . The expected exponential convergence is observed.

solve the shock better using the P_2 scheme. While the HWENO limiter is able to resolve the shock better than the Krivodonova limiter it does allow for more oscillations away from the shock. The WENO limiters result in an unstable evolution for schemes higher than P_2 . Ultimately, for the slow shock the HWENO and Krivodonova limiters perform comparably with the P_2 scheme and minimal benefit is seen when using higher order schemes with the Krivodonova limiter given the additional expense of the increased order.

Cylindrical Blast Wave

A standard test problem for GRMHD codes is known as the cylindrical blast wave [90, 92] where a magnetized fluid initially at rest in a constant magnetic field along the x -axis is evolved. The fluid obeys the ideal fluid equation of state,

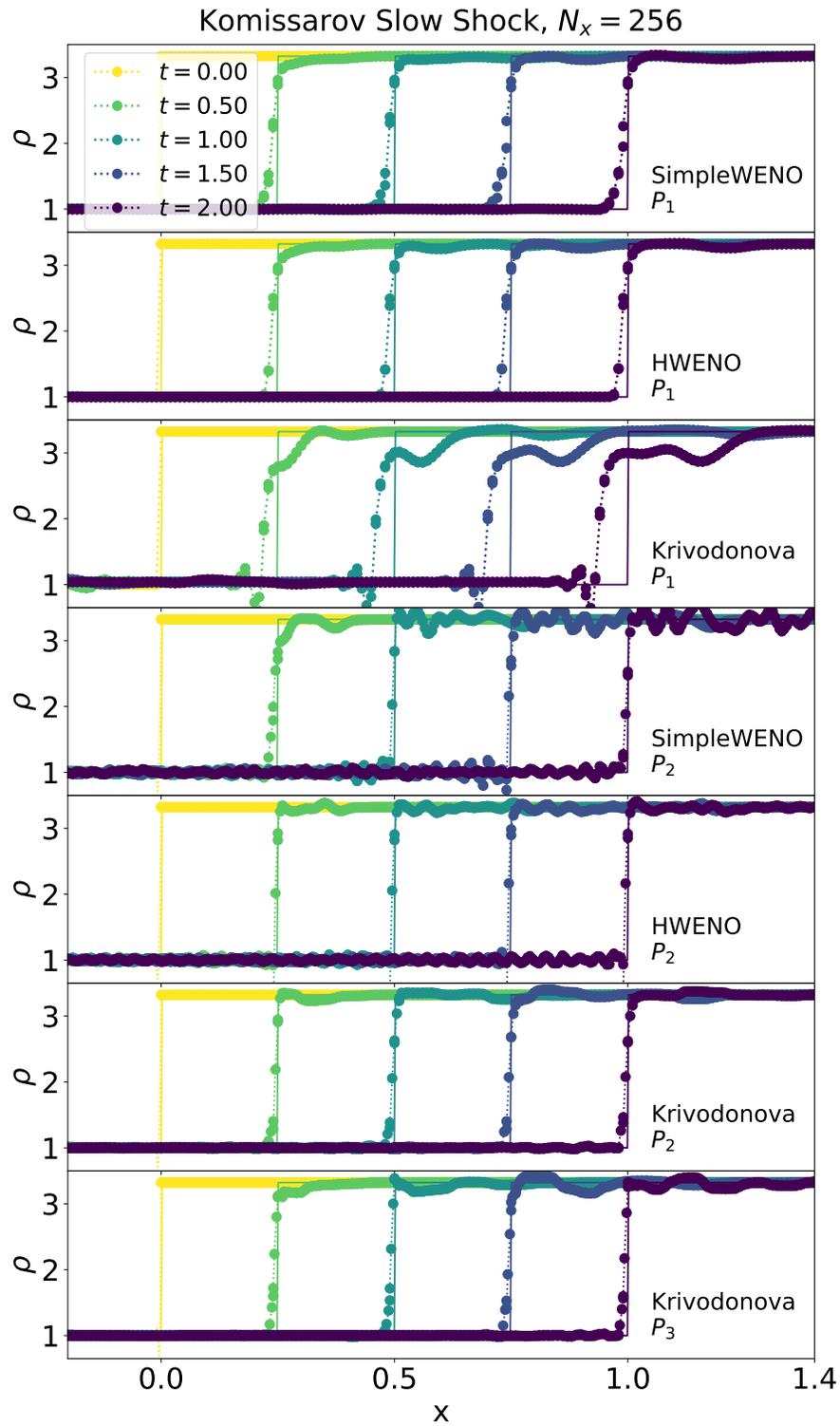


Figure 3.4: Evolution of the slow shock problem using P_1 and P_2 schemes using the simple WENO, HWENO, and Krivodonova limiters. Higher order schemes are unstable when using the WENO limiters.

| DG Scheme | N_x | $L_1(\mathcal{E}(v_z))$ | L_1 Order |
|-----------|-------|-------------------------|-------------|
| P_3 | 8 | 1.32233e-03 | 3.32 |
| | 16 | 8.64186e-05 | 3.94 |
| | 32 | 5.07122e-06 | 4.09 |
| P_5 | 4 | 1.01142e-04 | 5.52 |
| | 8 | 1.72526e-06 | 5.87 |
| | 16 | 2.54321e-08 | 6.08 |
| | 32 | 3.72441e-10 | 6.09 |
| P_7 | 1 | 1.28215e-02 | 6.63 |
| | 2 | 7.16386e-05 | 7.48 |
| | 4 | 3.38470e-07 | 7.73 |
| | 8 | 1.23088e-09 | 8.10 |
| | 16 | 4.41461e-12 | 8.12 |
| | 32 | 1.32994e-12 | 1.73 |
| P_9 | 2 | 6.67333e-07 | 10.57 |
| | 4 | 6.72052e-10 | 9.96 |
| | 8 | 1.17891e-11 | 5.83 |
| | 16 | 8.69128e-13 | 3.76 |

Table 3.1: The errors and local convergence order for large amplitude Alfvén wave evolutions. We observe the expected convergence rate except when the solution is underresolved because too few elements are used or when the error is no longer dominated by the truncation error of the DG scheme.

| State $x < 0$ | State $x > 0$ |
|------------------------|------------------------------|
| $\rho = 1$ | $\rho = 3.323$ |
| $p = 10$ | $P = 55.36$ |
| $u_i = (1.53, 0, 0)$ | $u_i = (0.9571, -0.6822, 0)$ |
| $B^i = (10, 18.28, 0)$ | $B^i = (10, 14.49, 0)$ |

Table 3.2: Initial conditions for the slow shock test.

(3.69), with $\gamma = 4/3$. The fluid begins in a cylindrically symmetric configuration, with hot, dense fluid in the region with cylindrical radius $r < 0.8$ surrounded by a cooler, less dense fluid in the region $r > 1.0$. The initial density ρ and pressure p

of the fluid are

$$\rho(r < 0.8) = 1.0 \times 10^{-2}, \quad (3.85)$$

$$\rho(r > 1.0) = 1.0 \times 10^{-4}, \quad (3.86)$$

$$p(r < 0.8) = 1.0, \quad (3.87)$$

$$p(r > 1.0) = 5.0 \times 10^{-4}. \quad (3.88)$$

In the region $0.8 \leq r \leq 1.0$, the solution transitions continuously and exponentially (i.e., transitions such that the logarithms of the pressure and density are linear functions of r). The fluid begins threaded with a uniform magnetic field with Cartesian components

$$(B^x, B^y, B^z) = (0.1, 0.0, 0.0). \quad (3.89)$$

The magnetic field causes the blast wave to expand nonaxisymmetrically.

We evolve the blast wave to time $t = 4.0$ on a grid of $128 \times 128 \times 1$ elements covering a cube of extent $[-6, 6] \times [-6, 6] \times [-6, 6]$. We apply periodic boundary conditions in all directions, since the explosion does not reach the outer boundary by $t = 4.0$. Figure 3.5 shows the logarithm of the rest-mass density at time $t = 4.0$, at the end of three evolutions that are identical except for the choice of limiter. The left column shows results using the Krivodonova limiter, with clearly visible artifacts along the Cartesian grid, especially when using the P_1 scheme. The middle column shows results using the simple WENO limiter, which is able to resolve the curved features of the blast wave much better with clear improvement in the P_2 case over the P_1 case. Finally, in the right column of Figure 3.5 we show results using the HWENO limiter. For the blast wave HWENO shows a clear improvement over simple WENO in its ability to resolve discontinuities and small features at higher order.

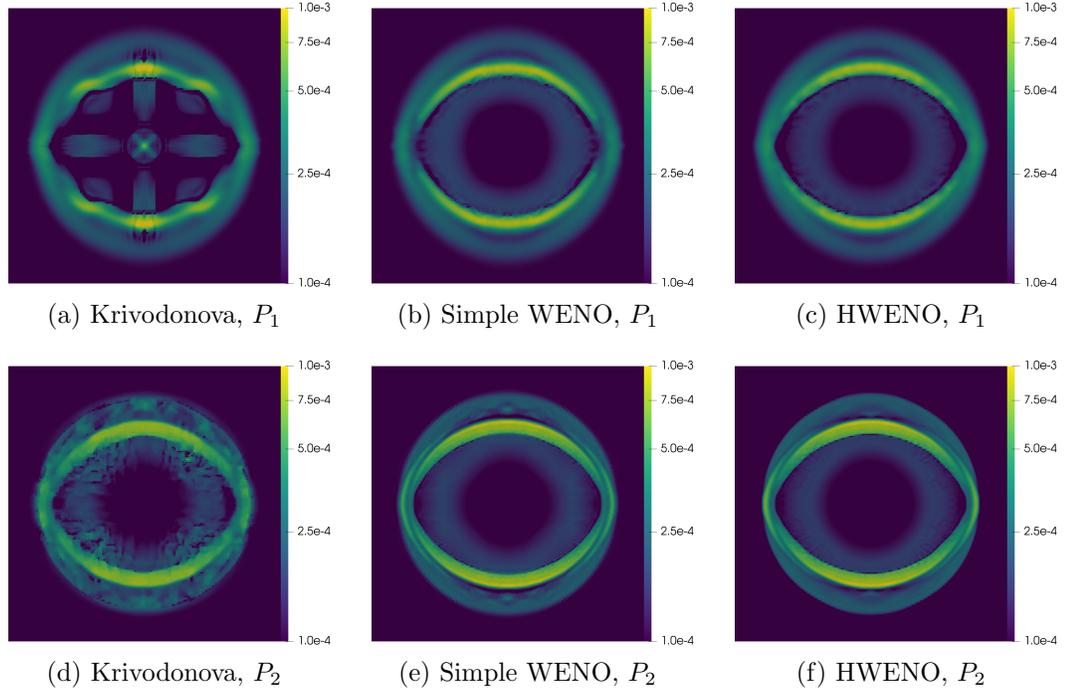


Figure 3.5: Cylindrical blast wave ρ at $t = 4$ comparing the Krivodonova, simple WENO, and HWENO limiters with P_1 (top row) and P_2 (bottom row).

Orszag-Tang vortex

The relativistic version of the Orszag-Tang vortex is a 2-dimensional test case for GRMHD systems (see, e.g., [93]). It describes the flow of an ideal fluid (3.69) with adiabatic index $5/3$. The initial conditions (and hence the states at later times) are periodic in both x and y with period 1. The initial conditions are:

$$\rho = \frac{25}{36\pi}, \quad (3.90)$$

$$p = \frac{5}{12\pi}, \quad (3.91)$$

$$v_x = -\frac{1}{2} \sin(2\pi y), \quad (3.92)$$

$$v_y = \frac{1}{2} \sin(2\pi x), \quad (3.93)$$

$$B_x = -\frac{1}{\sqrt{4\pi}} \sin(2\pi y), \quad (3.94)$$

$$B_y = \frac{1}{\sqrt{4\pi}} \sin(4\pi x), \quad (3.95)$$

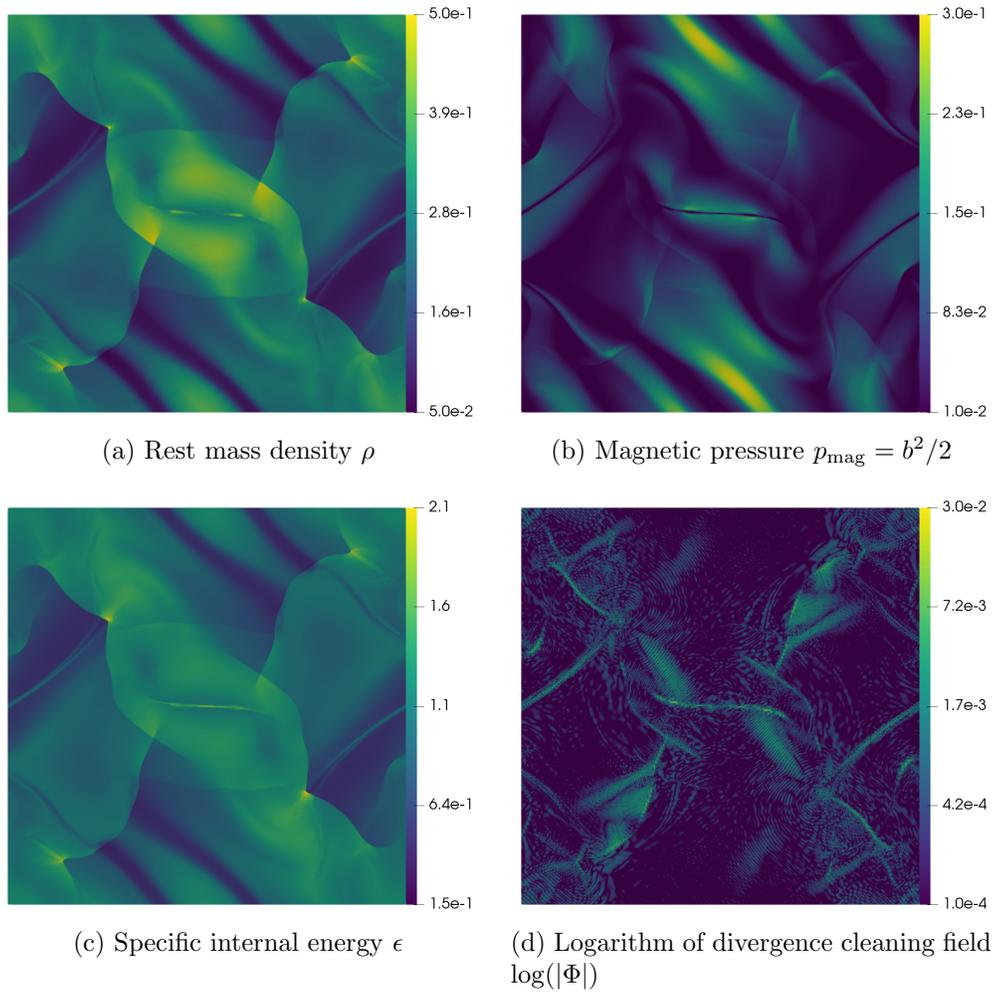


Figure 3.6: State of the Orszag-Tang vortex at time 1.0 evolved using a P_2 scheme supplemented by the HWENO limiter. Reading across: rest-mass density, magnetic pressure, specific internal energy, and the value of the divergence cleaning field.

where ρ is the rest-mass density, p the pressure, v_i the spatial velocity, and B_i the magnetic field.

As with the blast wave above, we evolve the system in 3 dimensions by having a single periodic element in the z direction. In the other two dimensions we use 256×256 linear elements. We evolve the system using a P_2 scheme supplemented by the HWENO limiter until time 1.0, at which point the state is as shown in Figure 3.6.

3.4.2 The Fishbone-Moncrief disk

The Fishbone-Moncrief disk [94] is an isentropic thick fluid disk orbiting a Kerr black hole. We provide an overview of the relevant equations and our notation in Appendix 3.A. We evolve the Fishbone-Moncrief disk on a cube of extent $[-40, 40] \times [2, 40] \times [-8, 8]$, with the mass of the black hole set to $M = 1$, dimensionless spin $\chi = 0.9375$, polytropic constant $K = 10^{-3}$, polytropic exponent $\gamma = 4/3$, inner edge radius $r_{\text{in}} = 6$, and max pressure radius $r_{\text{max}} = 12$. This configuration has a maximum rest mass density $\rho_{\text{max}} \approx 77$ and an orbital period $T \approx 247$.

To verify that we do not have slowly exponentially growing modes we evolve the disk to $t = 600$. In Figure 3.7 we show the L_2 error of the rest mass density as a function of time for three different resolutions. We use a P_2 scheme supplemented by the HWENO limiter and a third order time stepper. We observe that the errors grow early on and then reach a new equilibrium solution that appears to be stable over long time scales. We observe second order convergence.

In Figure 3.8 we plot the error in the rest mass density in the equatorial plane at $t = 600$ from the highest-resolution evolution. The disk is rotating counter clockwise, so the error build up on the left side of the disk is expected, especially

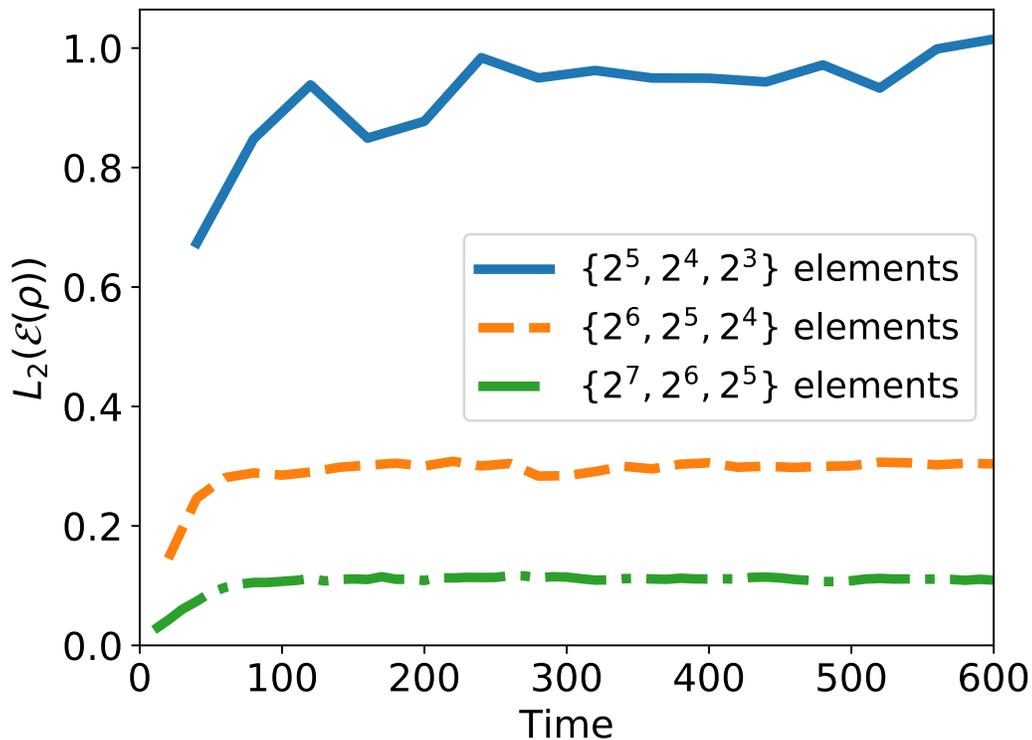


Figure 3.7: Error in the rest mass density ρ at three different resolutions for a Fishbone-Moncrief disk evolution using a P_2 scheme with the HWENO limiter. Second order convergence is observed.

given that we are imposing Dirichlet boundary conditions so the errors are unable to propagate off the grid. The disk evolves to a new equilibrium state where the left side is radially farther out than the exact solution. This is also the location of the largest errors, generally consistent with the difficulty of maintaining surfaces.

3.5 Acknowledgements

Charm++/Converse was developed by the Parallel Programming Laboratory in the Department of Computer Science at the University of Illinois at Urbana-Champaign.

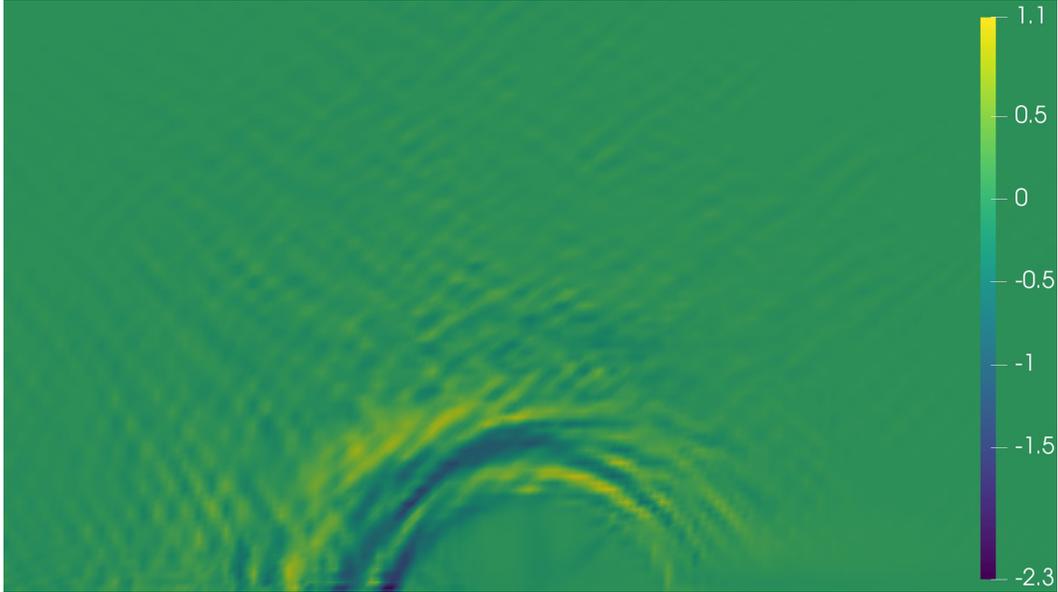


Figure 3.8: Error in the rest mass density ρ at $t = 600$ for a Fishbone-Moncrief disk evolution with $\{2^7, 2^6, 2^5\}$ elements in $\{x, y, z\}$, respectively, using a P_2 scheme with the HWENO limiter.

3.A Fishbone-Moncrief disk equations

The Fishbone-Moncrief solution [94] assumes a toroidal flow in Boyer-Lindquist coordinates (t, r, θ, ϕ) and so the 4-velocity of the fluid is given by

$$u^a = (u^t, 0, 0, u^\phi). \quad (3.96)$$

Ignoring self-gravity allows the fluid variables to be determined as functions of the metric. Following the treatment by Kozłowski et al. [95] (but using signature $(-, +, +, +)$) the solution is expressed in terms of the quantities

$$\Omega(r, \theta) = \frac{u^\phi}{u^t}, \quad (3.97)$$

$$W(r, \theta) = W_{\text{in}} - \int_{p_{\text{in}}}^p \frac{dp}{e + p}, \quad (3.98)$$

where Ω is the angular velocity, p is the fluid pressure, e is the energy density, and W is an auxiliary quantity interpreted in the Newtonian limit as the total (gravitational + centrifugal) potential. W_{in} and p_{in} are the potential and the

pressure at the radius of the inner edge r_{in} , i.e. the closest edge to the black hole.

Here we assume $p_{\text{in}} = 0$.

Using the above assumptions and definitions the solution can be written as

$$u^t = \sqrt{\frac{A}{2\Delta\Sigma} \left(1 + \sqrt{1 + \frac{4l^2\Delta\Sigma^2}{A^2 \sin^2 \theta}} \right)}, \quad (3.99)$$

$$\Omega = \frac{\Sigma}{A(u^t)^2 \sin^2 \theta} + \frac{2Mra}{A}, \quad (3.100)$$

$$u^\phi = \Omega u^t, \quad (3.101)$$

$$W = l\Omega - \ln u^t, \quad (3.102)$$

where

$$\Sigma = r^2 + a^2 \cos^2 \theta \quad (3.103)$$

$$\Delta = r^2 - 2Mr + a^2 \quad (3.104)$$

$$A = (r^2 + a^2)^2 - \Delta a^2 \sin^2 \theta \quad (3.105)$$

and $l = u_\phi u^t$ is the angular momentum per unit inertial mass, which parameterizes the disk. In deriving the solution, an integration constant has been chosen so that $W \rightarrow 0$ as $r \rightarrow \infty$, in accordance with the Newtonian limit. Note that, from its definition, equipotential contours coincide with isobaric contours.

Physically, the matter can fill each of the closed surfaces $W = \text{const}$, giving rise to an orbiting thick disk. For $W > 0$, all equipotentials are open, whereas for $W < 0$, some of them will be closed. Should a disk exist, the pressure reaches a maximum value on the equator at a coordinate radius r_{max} that is related to the angular momentum per unit inertial mass via

$$l = \frac{\sqrt{M} \left(r_{\text{max}}^{3/2} + a\sqrt{M} \right) \left(a^2 - 2a\sqrt{M}r_{\text{max}} + r_{\text{max}}^2 \right)}{2a\sqrt{M}r_{\text{max}}^3 + (r_{\text{max}} - 3M)r_{\text{max}}^2}. \quad (3.106)$$

Once W is determined, an equation of state is required in order to obtain the thermodynamic variables.

Since the flow is isentropic, the specific enthalpy can readily be obtained from the first and second laws of thermodynamics. Using

$$\frac{dp}{e+p} = \frac{dh}{h} \quad (3.107)$$

one finds that

$$h = h_{\text{in}} \exp(W_{\text{in}} - W), \quad (3.108)$$

The pressure can be obtained from a thermodynamic relation of the form $h = h(p)$. Here we assume a polytropic equation of state given by

$$p(\rho) = K\rho^\gamma, \quad (3.109)$$

where K is the polytropic constant and γ is the polytropic exponent.

Finally, the 4-velocity in Cartesian Kerr-Schild coordinates is given by

$$u_{\text{KS}}^a = u^t(1, -y\Omega, x\Omega, 0), \quad (3.110)$$

where u^t and Ω are now functions of the Kerr-Schild coordinates. The spatial velocity is obtained from its definition (3.11).

BIBLIOGRAPHY

- [1] B. P. Abbott et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016.
- [2] B. 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, 2018. arXiv:1811.12907.
- [3] B. P. Abbott et al. Multi-messenger observations of a binary neutron star merger. *The Astrophysical Journal*, 848(2):L12, oct 2017.
- [4] Prayush Kumar, Jonathan Blackman, Scott E. Field, Mark Scheel, Chad R. Galley, Michael Boyle, Lawrence E. Kidder, Harald P. Pfeiffer, Bela Szilagyi, and Saul A. Teukolsky. Constraining the parameters of GW150914 & GW170104 with numerical relativity surrogates, 2018. arXiv:1808.08004.
- [5] Nickolay Y. Gnedin, Vadim A. Semenov, and Andrey V. Kravtsov. Enforcing the Courant-Friedrichs-Lewy condition in explicitly conservative local time stepping schemes. *J Comp Phys*, 359:93–105, 2018.
- [6] Adrian Sandu and Emil M. Constantinescu. Multirate explicit Adams methods for time integration of conservation laws. *J Sci Comput*, 38(2):229–249, Feb 2009.
- [7] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J Comp Phys*, 53(3):484–512, 1984.
- [8] M. Berger. On conservation at grid interfaces. *SIAM J Numer Anal*, 24(5):967–984, 1987.

- [9] Lilia Krivodonova. An efficient local time-stepping scheme for solution of nonlinear conservation laws. *J Comp Phys*, 229(22):8537–8551, November 2010.
- [10] Gregor J. Gassner, Florian Hindenlang, and Claus-Dieter Munz. A Runge-Kutta based discontinuous Galerkin method with time accurate local time stepping. In *Adaptive High-Order Methods in Computational Fluid Dynamics*, pages 95–118. World Scientific, 2011.
- [11] M. Grote, M. Mehlin, and T. Mitkova. Runge-Kutta-based explicit local time-stepping methods for wave propagation. *SIAM J Sci Comput*, 37(2):A747–A775, 2015.
- [12] Pak-Wing Fok. A linearly fourth order multirate Runge-Kutta method with error control. *J Sci Comput*, 66(1):177–195, January 2016.
- [13] Michael Günther and Adrian Sandu. Multirate generalized additive Runge Kutta methods. *Numerische Mathematik*, 133(3):497–524, July 2016.
- [14] M. Almquist and M. Mehlin. Multilevel local time-stepping methods of Runge-Kutta-type for wave equations. *SIAM J Sci Comput*, 39(5):A2020–A2048, 2017.
- [15] M. Grote and T. Mitkova. Explicit local time-stepping methods for time-dependent wave propagation. *ArXiv e-prints*, May 2012.
- [16] Andrew R. Winters and David A. Kopriva. High-order local time stepping on moving DG spectral element meshes. *J Sci Comput*, 58(1):176–202, January 2014.

- [17] M. Grote, M. Mehlin, and S. Sauter. Convergence analysis of energy conserving explicit local time-stepping methods for the wave equation. *SIAM J Numer Anal*, 56(2):994–1021, 2018.
- [18] Emil M. Constantinescu and Adrian Sandu. Extrapolated multirate methods for differential equations with multiple time scales. *J Sci Comput*, 56(1):28–44, July 2013.
- [19] Svenja Schoeder, Martin Kronbichler, and Wolfgang A. Wall. Arbitrary high-order explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation. *J Sci Comput*, 76(2):969–1006, August 2018.
- [20] B. Chabaud and Q. Du. A hybrid implicit-explicit adaptive multirate numerical scheme for time-dependent equations. *J Sci Comput*, 51(1):135–157, April 2012.
- [21] Abdullah Demirel, Jens Niegemann, Kurt Busch, and Marlis Hochbruck. Efficient multiple time-stepping algorithms of higher order. *J Comp Phys*, 285:133–148, 2015.
- [22] Adrian Sandu and Michael Günther. A generalized-structure approach to additive Runge-Kutta methods. *SIAM J Numer Anal*, 53(1):17–42, 2015.
- [23] José Rafael Cavalcanti, Michael Dumbser, David da Motta-Marques, and Carlos Ruberto Frago Junior. A conservative finite volume scheme with time-accurate local time stepping for scalar transport on unstructured grids. *Advances in Water Resources*, 86:217–230, 2015.
- [24] L. F. Shampine. Conservation laws and the numerical solution of ODEs. *Computers & Mathematics with Applications*, 12(5, Part 2):1287–1296, 1986.

- [25] Pafnuty Chebyshev. Sur l'interpolation. In A. Markoff and N. Sonin, editors, *Oeuvres de P. L. Tchebychef*, volume 1, pages 541–560. l'Académie Impériale des Sciences, St. Petersburg, April 1899.
- [26] Saul A. Teukolsky. Formulation of discontinuous Galerkin methods for relativistic astrophysics. *J Comp Phys*, 312:333–356, 2016.
- [27] Lawrence E. Kidder, Scott E. Field, Francois Foucart, Erik Schnetter, Saul A. Teukolsky, Andy Bohn, Nils Deppe, Peter Diener, François Hébert, Jonas Lippuner, Jonah Miller, Christian D. Ott, Mark A. Scheel, and Trevor Vincent. SpECTRE: a task-based discontinuous Galerkin code for relativistic astrophysics. *J Comp Phys*, 335:84–114, 2017.
- [28] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. National topical meeting on mathematical models and computational techniques for analysis of nuclear systems, 10 1973.
- [29] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer-Verlag New York, New York, 2008.
- [30] B. Cockburn. Devising discontinuous Galerkin methods for non-linear hyperbolic conservation laws. *Journal of Computational and Applied Mathematics*, 128:187–204, March 2001.
- [31] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws v. multidimensional systems. *Journal of Computational Physics*, 141:199–224, April 1998.
- [32] Bernardo Cockburn. An introduction to the discontinuous galerkin method for convection-dominated problems. In Alfio Quarteroni, editor, *Advanced*

Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) held in Cetraro, Italy, June 23–28, 1997, pages 150–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

- [33] Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu. The development of discontinuous galerkin methods. In *Discontinuous Galerkin Methods*, pages 3–50, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [34] Scott E. Field, Jan S. Hesthaven, Stephen R. Lau, and Abdul H. Mroue. Discontinuous Galerkin method for the spherically reduced BSSN system with second-order operators. *Phys. Rev.*, D82:104051, 2010.
- [35] J. David Brown, Peter Diener, Scott E. Field, Jan S. Hesthaven, Frank Herrmann, Abdul H. Mroue, Olivier Sarbach, Erik Schnetter, Manuel Tiglio, and Michael Wagman. Numerical simulations with a first order BSSN formulation of Einstein’s field equations. *Phys. Rev.*, D85:084004, 2012.
- [36] Scott E. Field, Jan S. Hesthaven, and Stephen R. Lau. Discontinuous Galerkin method for computing gravitational waveforms from extreme mass ratio binaries. *Class. Quant. Grav.*, 26:165010, 2009.
- [37] Gerhard Zumbusch. Finite element, discontinuous Galerkin, and finite difference evolution schemes in spacetime. *Class. Quant. Grav.*, 26:175011, 2009.
- [38] Michael Dumbser, Federico Guercilena, Sven Köppel, Luciano Rezzolla, and Olindo Zanotti. Conformal and covariant Z4 formulation of the Einstein equations: strongly hyperbolic first-order reduction and solution with discontinuous Galerkin schemes. *Phys. Rev.*, D97(8):084053, 2018.

- [39] David Radice and Luciano Rezzolla. Discontinuous Galerkin methods for general-relativistic hydrodynamics: formulation and application to spherically symmetric spacetimes. *Phys. Rev.*, D84:024010, 2011.
- [40] Philip Mocz, Mark Vogelsberger, Debora Sijacki, Rüdiger Pakmor, and Lars Hernquist. A discontinuous Galerkin method for solving the fluid and magnetohydrodynamic equations in astrophysical simulations. *Mon. Not. Roy. Astron. Soc.*, 437(1):397–414, 2014.
- [41] Olindo Zanotti, Michael Dumbser, Arturo Hidalgo, and Dinshaw Balsara. An ADER-WENO finite volume AMR code for astrophysics. *ASP Conf. Ser.*, 488:285–290, 2014.
- [42] Eirik Endeve, Cory D. Hauck, Yulong Xing, and Anthony Mezzacappa. Bound-preserving discontinuous Galerkin methods for conservative phase space advection in curvilinear coordinates. *J. Comput. Phys.*, 287:151–183, 2015.
- [43] Kevin Schaal, Andreas Bauer, Praveen Chandrashekar, Rüdiger Pakmor, Christian Klingenberg, and Volker Springel. Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement. *Mon. Not. Roy. Astron. Soc.*, 453(4):4278–4300, 2015.
- [44] 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.*, D94(8):084004, 2016.
- [45] Olindo Zanotti, Francesco Fambri, and Michael Dumbser. Solving the relativistic magnetohydrodynamics equations with ADER discontinuous Galerkin

- methods, a posteriori subcell limiting and adaptive mesh refinement. *Mon. Not. Roy. Astron. Soc.*, 452(3):3010–3029, 2015.
- [46] Michael Dumbser, Francesco Fambri, Maurizio Tavelli, Michael Bader, and Tobias Weinzierl. Efficient implementation of ADER discontinuous Galerkin schemes for a scalable hyperbolic PDE engine. *Axioms*, 7(3), August 2018.
- [47] Francesco Fambri, Michael Dumbser, Sven Köppel, Luciano Rezzolla, and Olindo Zanotti. ADER discontinuous Galerkin schemes for general-relativistic ideal magnetohydrodynamics. *Mon. Not. Roy. Astron. Soc.*, 477(4):4543–4564, 2018.
- [48] Olindo Zanotti, Francesco Fambri, Michael Dumbser, and Arturo Hidalgo. Space-time adaptive ADER discontinuous Galerkin finite element schemes with a posteriori sub-cell finite volume limiting. *Computers & Fluids*, 118:204–224, 2015.
- [49] Olindo Zanotti and Michael Dumbser. Efficient conservative ADER schemes based on WENO reconstruction and space-time predictor in primitive variables. *Computational Astrophysics and Cosmology*, 3(1):1, January 2016.
- [50] Jonah M. Miller and Erik Schnetter. An operator-based local discontinuous Galerkin method compatible with the BSSN formulation of the Einstein equations. *Class. Quant. Grav.*, 34(1):015003, 2017.
- [51] Thomas W. Baumgarte and Stuart L. Shapiro. *Numerical Relativity: Solving Einstein’s Equations on the Computer*. Cambridge University Press, 2010.
- [52] L. Rezzolla and O. Zanotti. *Relativistic Hydrodynamics*. Oxford University Press, September 2013.

- [53] Luis Antón, Olindo Zanotti, Juan A. Miralles, José M. Martí, José M. Ibáñez, José A. Font, and Jos´ A. Pons. Numerical 3+1 general relativistic magneto-hydrodynamics: A local characteristic approach. *The Astrophysical Journal*, 637:296–312, January 2006.
- [54] Jose A. Font. Numerical hydrodynamics and magnetohydrodynamics in general relativity. *Living Rev. Rel.*, 11:7, 2008.
- [55] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wengenber. Hyperbolic divergence cleaning for the MHD equations. *Journal of Computational Physics*, 175:645–673, January 2002.
- [56] David Hilditch and Andreas Schoepe. Hyperbolicity of divergence cleaning and vector potential formulations of GRMHD, 2018.
- [57] C. F. Gammie, J. C. McKinney, and G. Tóth. HARM: a numerical scheme for general relativistic magnetohydrodynamics. *Astrophys. J.*, 589:444–457, May 2003.
- [58] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer Science+Business Media B.V., 2009.
- [59] 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.
- [60] Alexander Heinecke, Hans Pabst, and Greg Henry. LIBXSMM: a high performance library for small matrix multiplications. In *SC’15: The International Conference for High Performance Computing, Networking, Storage and Analysis*, Austin, Texas, 2015.

- [61] Alexander Heinecke, Greg Henry, Maxwell Hutchinson, and Hans Pabst. LIBXSMM: accelerating small matrix multiplications by runtime code generation. In *SC'16: The International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, Utah, 2016.
- [62] Alexander Heinecke, Evangelos Georganas, Kunal Banerjee, Dhiraj Kalmakar, Narayanan Sundaram, Anand Venkat, Greg Henry, and Hans Pabst. Understanding the performance of small convolution operations for cnn on intel architecture. In *SC'17: The International Conference for High Performance Computing, Networking, Storage, and Analysis*, Denver, Colorado, 2017.
- [63] Evangelos Georganas, Sasikanth Avancha, Kunal Banerjee, Dhiraj Kalmakar, Greg Henry, Hans Pabst, and Alexander Heinecke. Anatomy of high-performance deep learning convolutions on simd architectures. In *SC'18: The International Conference for High Performance Computing, Networking, Storage, and Analysis*, Dallas, Texas, 2018.
- [64] K. Iglberger, G. Hager, J. Treibig, and U. Rde. Expression templates revisited: A performance analysis of current methodologies. *SIAM Journal on Scientific Computing*, 34(2):C42–C69, 2012.
- [65] K. Iglberger, G. Hager, J. Treibig, and U. Rde. High performance smart expression template math libraries. In *2012 International Conference on High Performance Computing Simulation (HPCS)*, pages 367–373, July 2012.
- [66] jemalloc. <https://github.com/jemalloc/jemalloc>, 2011.
- [67] tcmalloc. <https://github.com/gperftools/gperftools>, 2011.
- [68] R. D. Nair, S. J. Thomas, and R. D. Loft. A discontinuous Galerkin transport scheme on the cubed sphere. *Monthly Weather Review*, 133:814, 2005.

- [69] Mark A. Scheel, Lawrence E. Kidder, Lee Lindblom, Harald P. Pfeiffer, and Saul A. Teukolsky. Toward stable 3-D numerical evolutions of black hole space-times. *Phys. Rev.*, D66:124005, 2002.
- [70] 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.
- [71] Lilia Krivodonova. Limiters for high-order discontinuous Galerkin methods. *Journal of Computational Physics*, 226(1):879 – 896, 2007.
- [72] X. Zhong and C.-W. Shu. A simple weighted essentially nonoscillatory limiter for Runge-Kutta discontinuous Galerkin methods. *Journal of Computational Physics*, 232:397–415, January 2013.
- [73] J. Zhu, X. Zhong, C.-W. Shu, and J. Qiu. Runge-Kutta discontinuous Galerkin method with a simple and compact Hermite WENO limiter. *Communications in Computational Physics*, 19:944–969, April 2016.
- [74] B. Cockburn, S.-Y. Lin, and C.-W. Shu. TVB Runge Kutta local projection discontinuous Galerkin finite element method for conservation laws iii: One-dimensional systems. *Journal of Computational Physics*, 84:90–113, September 1989.
- [75] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. iv. the multidimensional case. *Mathematics of Computation*, 54:545–581, April 1990.
- [76] M. Dumbser and M. Käser. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. *Journal of Computational Physics*, 221:693–723, February 2007.

- [77] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, and J.E. Flaherty. Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. *Applied Numerical Mathematics*, 48(3):323 – 338, 2004.
- [78] W. Throwe and S. A. Teukolsky. A high-order, conservative integrator with local time-stepping, November 2018.
- [79] V. V. Rusanov. Calculation of interaction of non-steady shock waves with obstacles. *J. Comput. Math. Phys. USSR*, 1:267–279, 1961.
- [80] A. Harten, P. Lax, and B. Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61, 1983.
- [81] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer Berlin Heidelberg, 2009.
- [82] S. Davis. Simplified second-order Godunov-type methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445–473, 1988.
- [83] Daniel M. Siegel, Philipp Mösta, Dhruv Desai, and Samantha Wu. Recovery schemes for primitive variables in general-relativistic magnetohydrodynamics. *Astrophys. J.*, 859(1):71, 2018.
- [84] W. Newman and N. Hamlin. Primitive variable determination in conservative relativistic magnetohydrodynamic simulations. *SIAM Journal on Scientific Computing*, 36(4):B661–B683, 2014.
- [85] Carlos Palenzuela, Steven L. Liebling, David Neilsen, Luis Lehner, O. L. Caballero, Evan O’Connor, and Matthew Anderson. Effects of the microphysical equation of state in the mergers of magnetized neutron stars with neutrino cooling. *Phys. Rev.*, D92(4):044045, 2015.

- [86] Filippo Galeazzi, Wolfgang Kastaun, Luciano Rezzolla, and José A. Font. Implementation of a simplified approach to radiative transfer in general relativity. *Phys. Rev.*, D88:064009, 2013.
- [87] Francois Foucart. *Numerical Studies Of Black Hole-Neutron Star Binaries*. PhD thesis, Cornell University, 2011.
- [88] Curran D. Muhlberger, Fatemeh Hossein Nouri, Matthew D. Duez, Francois Foucart, Lawrence E. Kidder, Christian D. Ott, Mark A. Scheel, Béla Szilágyi, and Saul A. Teukolsky. Magnetic effects on the low- $T/|W|$ instability in differentially rotating neutron stars. *Phys. Rev.*, D90(10):104014, 2014.
- [89] Laxmikant V Kale and Sanjeev Krishnan. Charm++: Parallel programming with message-driven objects. In Gregory V. Wilson and Paul Lu, editors, *Parallel programming using C++*, pages 175–213. The MIT Press, 1996.
- [90] L. Del Zanna, O. Zanotti, N. Bucciantini, and P. Londrillo. ECHO: an Eulerian conservative high order scheme for general relativistic magnetohydrodynamics and magnetodynamics. *Astron. Astrophys.*, 473:11–30, 2007.
- [91] S. S. Komissarov. A Godunov-type scheme for relativistic magnetohydrodynamics. *Mon. Not. R. Astron. Soc.*, 303:343–366, February 1999.
- [92] T. Leismann, L. Antón, M. A. Aloy, E. Müller, J. M. Martí, J. A. Miralles, and J. M. Ibáñez. Relativistic MHD simulations of extragalactic jets. *Astron. Astrophys.*, 436:503–526, June 2005.
- [93] Kris Beckwith and James M. Stone. A second order Godunov method for multidimensional relativistic magnetohydrodynamics. *Astrophys. J. Suppl.*, 193:6, 2011.

- [94] L. G. Fishbone and V. Moncrief. Relativistic fluid disks in orbit around Kerr black holes. *Astrophysical Journal*, 207:962–976, August 1976.
- [95] M. Kozłowski, M. Jaroszynski, and M. A. Abramowicz. The analytic theory of fluid disks orbiting the Kerr black hole. *Astronomy and Astrophysics*, 63:209–220, February 1978.