Cholla-MHD: An Exascale-Capable Magnetohydrodynamic Extension to the Cholla Astrophysical Simulation Code

by

Robert V. Caddy

Bachelor of Science, Purdue University, 2016

Master of Science, Bowling Green State University, 2018

Submitted to the Graduate Faculty of

the Dietrich School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2024

UNIVERSITY OF PITTSBURGH

DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Robert V. Caddy

It was defended on

April 1st 2024

and approved by

Evan E. Schneider, University of Pittsburgh, Department of Physics and Astronomy

Rachel Bezanson, University of Pittsburgh, Department of Physics and Astronomy

Carles Badenes, University of Pittsburgh, Department of Physics and Astronomy

Ayres Freitas, University of Pittsburgh, Department of Physics and Astronomy

Tiziana Di Matteo, Carnegie Mellon University, Department of Physics

Cholla-MHD: An Exascale-Capable Magnetohydrodynamic Extension to the Cholla Astrophysical Simulation Code

Robert V. Caddy, PhD

University of Pittsburgh, 2024

We present an extension of the massively parallel, GPU native, astrophysical hydrodynamics code Cholla to magnetohydrodynamics (MHD). Cholla solves the ideal MHD equations in their Eulerian form on a static Cartesian mesh utilizing the Van Leer + Constrained Transport integrator, the HLLD Riemann solver, and reconstruction methods at second and third order. Cholla's MHD module can perform ≈ 260 million cell updates per GPU-second on an NVIDIA A100 while using the HLLD Riemann solver and second order reconstruction. The inherently parallel nature of GPUs combined with increased memory in new hardware allows Cholla's MHD module to perform simulations with resolutions $\sim 500^3$ cells on a single high end GPU (e.g. an NVIDIA A100 with 80GB of memory). We employ GPU direct MPI to attain excellent weak scaling on the exascale supercomputer *Frontier*, while using 74,088 GPUs and simulating a total grid size of over 7.2 trillion cells. A suite of test problems highlights the accuracy of Cholla's MHD module and demonstrates that zero magnetic divergence in solutions is maintained to round off error. We also present new testing and continuous integration tools using GoogleTest, GitHub Actions, and Jenkins that have made development more robust and accurate and ensure reliability in the future.

Table of Contents

List of Tables

List of Figures

Preface

Acknowledgements

RVC thanks Alwin Mao, Helena Richie, Orlando Warren, Kyle Felker, and Seth Cook for many helpful discussions. Special thanks to Daniel Perrefort at the Center for Research Computing for his help in setting up and maintaining the Jenkins environment for our automated testing. This research was supported in part by the University of Pittsburgh Center for Research Computing, RRID:SCR_022735, through the resources provided. Specifically, this work used the H2P cluster, which is supported by NSF award number OAC-2117681. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725, using Frontier CAAR allocations CSC380 and AST181. E.E.S. acknowledges support from NASA TCAN grant 80NSSC21K0271, NASA ATP grant 80NSSC22K0720, StScI grant HST-AR-16633.001-A, and the David and Lucile Packard Foundation (grant no. 2022-74680).

With the exceptions of Chapter 1 and the Appendix, this dissertation has been submitted for publication in the Astrophysical Journal. Reproduced with permission of the copyright owners.

1.0 Background

This dissertation is primarily concerned with two topics: adding magnetohydrodynamics (MHD) to the hydrodynamics code Cholla[98] and the implementation of several different scientific software best practices into Cholla. With these changes, Cholla now has the capacity to more accurately model astrophysical phenomenons such as galactic outflows. The advent of exascale supercomputers such as *Frontier* combined with high-performing simulation codes like Cholla will allow us to model the universe in unprecedented detail.

1.1 Numerical Magnetohydrodynamics in Astrophysics

Numerical modeling has become a critical and all-encompassing tool in modern astrophysics. Complex numerical models underpin nearly all modern astrophysics in theory and observation alike. Simulations are an especially powerful tool for theoretical astrophysics as it is often impossible to conduct the necessary experiments in a laboratory. The simulation codes in astrophysics commonly model hydrodynamics, often with additional physics such as radiation, conduction, magnetic fields, cosmic rays, general relativity, self gravity, star formation, stellar feedback, AGN feedback, etc. As such a wide variety of codes and methods have been developed to answer questions ranging from the dynamics of stars to cosmological structure.

1.1.1 Supercomputers and the Advent of Accelerators

Astrophysical hydrodynamics codes are complex and computationally demanding, especially with the inclusion of additional physics such as radiative transfer. These codes typically require computers with performance in the hundreds of teraFLOPS or higher for simulations to run in reasonable amounts of times and cutting edge simulations require hundreds or thousands of petaFLOPS. To meet these high computational demands we must

rely on supercomputers. Until the early 2000's the primary ways of making supercomputers faster was to increase clock speeds and increase the number of processors. However, these ever-higher clock speeds also pushed the power density to over $10W/cm^2$ by the late 1990's and 10 times that by 2010[63, 66]. These high power densities, several times higher than a hot plate, forced a transition in the early 2000's to multi-core processors and then to accelerators such as GPUs due to their higher efficiency. These accelerators have become ubiquitous in the world of supercomputing. At the time of writing, all but one of the top 10 fastest supercomputers in the world get the majority of their performance from accelerators in the form of GPUs[1] and the one exception, *Fugaku*, uses custom CPUs that essentially have integrated accelerators.

GPUs function significantly differently than CPUs. Where a modern CPU has 4-128 cores, modern HPC (High Performance Computing) GPUs have well over 10,000 cores. These cores however are much smaller and simpler than their CPU counterparts, mostly intended for performing many mathematical operations in parallel. GPU cores also must operate in lock step in groups of 32 or 64[2] in a Single Instruction/Multiple Data paradigm where each core performs the same operation on a different set of data. This makes GPUs ideal for performing grid-based hydrodynamical simulations, where we need to perform the same operations on every one of billions, or even trillions, of cells. GPUs are especially well suited to MHD codes due to the computational cost and high arithmetic intensity of such codes.

1.1.2 Eulerian vs. Lagrangian Methods

The two most common numerical techniques in astrophysics for solving compressible fluid equations are Smoothed Particle Hydrodynamics (SPH) and Eulerian grid codes. SPH is a Lagrangian method that discretizes the fluid in mass in the form of particles; these fluid particles are then evolved in their own rest frame. Eulerian codes instead discretize the fluid in space with many cells that are assigned the average properties of the fluid within that cell. The flux of the various fluid properties is then computed through each cell interface and those fluxes are used to evolve the fluid values within each cell. SPH

[1] `https://www.top500.org`

[2] The exact number of threads that have to operate in lock step depends on the GPU vendor

does an excellent job of automatically allocating resolution in high density regions where it is usually needed most, is less susceptible to round-off error in high mach flows, and more easily maintains hydrostatic equilibrium. To achieve this same adaptive resolution, grid-based codes must utilize complex AMR (adaptive mesh refinement) schemes that lead to additional complexity. Grid-based codes, however, do a better job of modeling shocks and conserving fluid fluxes across the domain[108]. Since many astrophysical phenomena exhibit strong and complex shocks that drive much of the dynamics, grid-based codes are an excellent choice for astrophysical modeling. Their regular data structures also make load balancing and efficiently using many-core architectures much easier.

Grid-based codes are also especially adept at MHD. Their regular structure is conducive to the field nature of magnetic fields and regular grids make divergence-free methods like constrained transport tractable to implement. Hybrid SPH and grid-based methods also exist, notably in the GIZMO and AREPO codes [59, 104]. These methods do accomplish many of their goals of combining the strengths of both Eulerian and Lagrangian codes. However, they are very complex with significant computational overhead. Additionally, their often non-uniform grid structures often make constrained transport intractable to implement.

1.1.3 Introduction to Finite-Volume Methods

1.1.3.1 Hydrodynamics

Eulerian finite-volume methods are based on discretizing the simulation volume into elements called cells. Each cell is assigned the average value for the conserved fluid quantities within the volume of that cell. These quantities, density(ρ), momentum density($\rho\vec{v}$), energy density(E), and magnetic field density($\vec{B}$) are then evolved. In hydrodynamics the energy is the sum of the kinetic and internal energies $E = \rho\left(\frac{1}{2}\vec{v}^2 + e\right)$ where e is the internal energy. The Euler equations written in terms of the conserved variables are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{1}$$

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + P\boldsymbol{I}) = 0 \tag{2}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + P)\boldsymbol{v}) = 0 \tag{3}$$

where P is the pressure and $\boldsymbol{I}$ is the identity tensor. When coupled with an equation of state these equations can be solved; we use the ideal gas equation of state $P = \rho(\gamma - 1)e$ where γ is the adiabatic index and ϵ is the internal energy density. Equation 1 describes the conservation of mass, equation 2 describes the conservation of momentum, equation 3 describes the conservation of energy.

In practice, these equations are used in their vector form, where $\boldsymbol{U}$ and $\boldsymbol{W}$ are the conserved and primitive variables respectively:

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ v_x \\ v_y \\ v_z \\ E \end{bmatrix}, \boldsymbol{W} = \begin{bmatrix} \rho \\ v_x \\ v_y \\ v_z \\ P \end{bmatrix}. \tag{4}$$

The conservation equations, in Cartesian coordinates, can then be rewritten as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} + \frac{\partial \boldsymbol{H}}{\partial z} = 0 \tag{5}$$

where $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ are the vectors of fluxes in the x, y, and z direction respectively and are given by

$$\boldsymbol{F} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P \\ \rho v_x v_y \\ \rho v_x v_z \\ v_x \left(E + P\right) \end{bmatrix} \tag{6}$$

$$\boldsymbol{G} = \begin{bmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + P \\ \rho v_y v_z \\ v_y \left(E + P\right) \end{bmatrix} \tag{7}$$

$$\boldsymbol{H} = \begin{bmatrix} \rho v_z \\ \rho v_z v_x \\ \rho v_z v_y \\ \rho v_z^2 + P \\ v_z \left(E + P\right) \end{bmatrix}. \tag{8}$$

The arrangement of these quantities on a given cell can be seen in Figure 1.

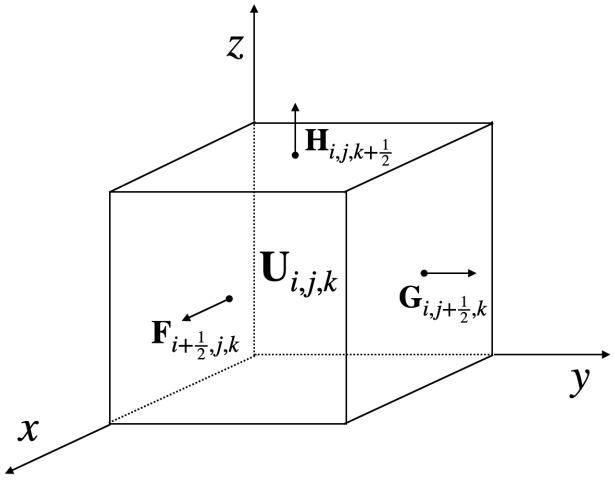

Figure 1: A single cell in a hydrodynamic finite-volume simulation. Average values of the conserved variables, U, are cell-centered while the fluxes are centered on the faces of the cell.

The fluxes are computed using a Riemann solver. The Riemann problem is an initial conditions problem where a system of conservation equations can be solved exactly if the initial conditions are a piecewise constant function with a single discontinuity. This exact situation is found at the interface between any two cells. A variety of Riemann solvers exist.

Cholla implements the Exact[109], Roe[94], HLL (Harten–Lax–van Leer)[54], and HLLC (Harten–Lax–van Leer contact)[5] Riemann solvers for hydrodynamics.

These fluxes can then be combined to update the conserved quantities using equation

$$\begin{aligned} \boldsymbol{U}_{i,j,k}^{n+1} = \boldsymbol{U}_{i,j,k}^{n} &- \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i+1/2,j,k}^{n} - \boldsymbol{F}_{i-1/2,j,k}^{n}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\boldsymbol{G}_{i+1/2,j,k}^{n} - \boldsymbol{G}_{i-1/2,j,k}^{n}\right) \\ &- \frac{\Delta t}{\Delta z}\left(\boldsymbol{H}_{i+1/2,j,k}^{n} - \boldsymbol{H}_{i-1/2,j,k}^{n}\right). \end{aligned} \tag{9}$$

where the n superscript indicates the time step and the i, j, k subscripts indicate the cell position; the $\pm 1/2$ terms indicate the location of the cell face at $\pm 1/2$ a cell width.

1.1.3.2 Magnetohydrodynamics

The ideal MHD formalism is much the same as the hydrodynamic formalism with the inclusion of the magnetic fields and the induction equation (equation 13) which serves as the magnetic field's conservation equation and encodes the divergence-free condition. Ideal MHD makes several approximations:

- Local thermodynamic equilibrium
- In the plasma there is a local, instantaneous relation between electric field and current density
- The plasma is electrically neutral.
- no electrical resistance, i.e. perfect conductivity
- The plasma is fully ionized

Details of the method implemented can be found in Section 3.1. The ideal MHD Euler equations are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{10}$$

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} - \boldsymbol{B} \otimes \boldsymbol{B} + P_T \boldsymbol{I}) = 0 \tag{11}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + P_T)\boldsymbol{v} + \boldsymbol{B}(\boldsymbol{B} \cdot \boldsymbol{v})) = 0 \tag{12}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} - \nabla \times (\boldsymbol{v} \times \boldsymbol{B}) = 0 \tag{13}$$

where $P_T \equiv P + \frac{1}{2}(\boldsymbol{B} \cdot \boldsymbol{B})$ is the total pressure, and E is the total energy per unit volume $E \equiv \epsilon + \frac{1}{2}\rho(\boldsymbol{v} \cdot \boldsymbol{v}) + \frac{1}{2}(\boldsymbol{B} \cdot \boldsymbol{B})$. We adopt units in which the magnetic permeability $\mu_0 = 1$.

Rewriting these equations into their vector forms:

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ v_x \\ v_y \\ v_z \\ E \\ B_x \\ B_y \\ B_z \end{bmatrix}, \boldsymbol{W} = \begin{bmatrix} \rho \\ v_x \\ v_y \\ v_z \\ P \\ B_x \\ B_y \\ B_z \end{bmatrix}. \tag{14}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} + \frac{\partial \boldsymbol{H}}{\partial z} = 0. \tag{15}$$

Where the fluxes are:

$$\boldsymbol{F} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P_T - B_x^2 \\ \rho v_x v_y - B_x B_y \\ \rho v_x v_z - B_x B_z \\ v_x \left(E + p_T\right) - B_x \left(\boldsymbol{v} \cdot \boldsymbol{B}\right) \\ 0 \\ B_y v_x - B_x v_y \\ B_z v_x - B_x v_z \end{bmatrix} \tag{16}$$

$$\boldsymbol{G} = \begin{bmatrix} \rho v_y \\ \rho v_y v_x - B_y B_x \\ \rho v_y^2 + P_T - B_y^2 \\ \rho v_y v_z - B_y B_z \\ v_y \left(E + P_T\right) - B_y \left(\boldsymbol{v} \cdot \boldsymbol{B}\right) \\ B_x v_y - B_y v_x \\ 0 \\ B_z v_y - B_y v_z \end{bmatrix} \tag{17}$$

$$\boldsymbol{H} = \begin{bmatrix} \rho v_z \\ \rho v_z v_x - B_z B_x \\ \rho v_z v_y - B_z B_y \\ \rho v_z^2 + P_T - B_z^2 \\ v_z \left(E + P_T\right) - B_z \left(\boldsymbol{v} \cdot \boldsymbol{B}\right) \\ B_x v_z - B_z v_x \\ B_y v_z - B_z v_y \\ 0 \end{bmatrix}. \tag{18}$$

The induction equation requires that the magnetic field be divergence free, i.e. the Universe does not contain magnetic monopoles. However, not all numerical methods for evolving the MHD equations maintain this constraint. In many particle-based schemes, for example, magnetic divergence is generated and is removed at each step, a method commonly known as divergence cleaning [27]. Divergence cleaning is popular because it is simple and couples well to particle-based methods – it essentially functions by computing the divergence regularly and subtracting it away. Divergence cleaning is also computationally cheaper than numerical methods that maintain a divergence-free solution by construction, but typically leads to divergence errors on the level of a few percent [27, 85, 114]. While this error is small it could lead to unphysical solutions. Divergence cleaning is also commonly used for grid-based codes. However, a more accurate method for evolving the magnetic fields exists for grid based codes.

The other primary method for evolving the magnetic field is constrained transport (CT).

Constrained transport is formally divergence free by construction, and when implemented numerically it typically results in divergence errors on the order of machine round off error [31, 38, 106, 105, 123, 1]. This is accomplished by tracking magnetic fields on a staggered, face centered grid rather than using cell-centered averages. These face centered values are used in conjunction with Riemann fluxes to calculate edge centered electric fields, and those electric fields are used to update the magnetic field; this arrangement can be seen in Figure 2. Thus, the trade-off for a divergence-free method is significant additional algorithmic complexity and associated computational expense.

The edge centered electric fields are computed using the magnetic fluxes from the Riemann solver and their derivatives. An example of this for the electric field in the z direction is shown in Equation 19

$$\begin{aligned}\mathcal{E}_{z,i-1/2,j-1/2,k} = \frac{1}{4}\left(\mathcal{E}_{z,i-1/2,j,k} + \mathcal{E}_{z,i,j-1/2,k} + \mathcal{E}_{z,i-1/2,j-1,k} + \mathcal{E}_{z,i-1,j-1/2,k}\right) \\ + \frac{\Delta y}{8}\left(\left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1/2,j-1/4,k} + \left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1/2,j-3/4,k}\right) \\ + \frac{\Delta x}{8}\left(\left(\frac{\partial \mathcal{E}_z}{\partial x}\right)_{i-1/4,j-1/2,k} + \left(\frac{\partial \mathcal{E}_z}{\partial x}\right)_{i-3/4,j-1/2,k}\right).\end{aligned} \tag{19}$$

These electric fields are then used to update the magnetic fields through the use of Faraday's Law ($\nabla \times \vec{E} = -\partial_t \vec{B}$). Since the divergence of a curl is zero this means that as long as the initial magnetic fields are divergence free then the update to them will maintain that divergence free status.

$$\begin{aligned}B^{n+1}_{x,i-1/2,j,k} = B^{n}_{x,i-1/2,j,k} + \frac{\Delta t}{\Delta z}\left(\mathcal{E}^n_{y,i-1/2,j,k+1/2} - \mathcal{E}^n_{y,i-1/2,j,k-1/2}\right) \\ - \frac{\Delta t}{\Delta y}\left(\mathcal{E}^n_{z,i-1/2,j+1/2,k} - \mathcal{E}^n_{z,i-1/2,j-1/2,k}\right)\end{aligned} \tag{20}$$

$$\begin{aligned}B^{n+1}_{y,i,j-1/2,k} = B^{n}_{y,i,j-1/2,k} + \frac{\Delta t}{\Delta x}\left(\mathcal{E}^n_{z,i+1/2,j-1/2,k} - \mathcal{E}^n_{z,i-1/2,j-1/2,k}\right) \\ - \frac{\Delta t}{\Delta z}\left(\mathcal{E}^n_{x,i,j-1/2,k+1/2} - \mathcal{E}^n_{x,i,j-1/2,k-1/2}\right)\end{aligned} \tag{21}$$

$$\begin{aligned}B^{n+1}_{z,i,j,k-1/2} = B^{n}_{z,i-1/2,j,k} + \frac{\Delta t}{\Delta y}\left(\mathcal{E}^n_{x,i,j+1/2,k-1/2} - \mathcal{E}^n_{x,i,j-1/2,k-1/2}\right) \\ - \frac{\Delta t}{\Delta x}\left(\mathcal{E}^n_{y,i+1/2,j,k-1/2} - \mathcal{E}^n_{y,i-1/2,j,k-1/2}\right).\end{aligned} \tag{22}$$

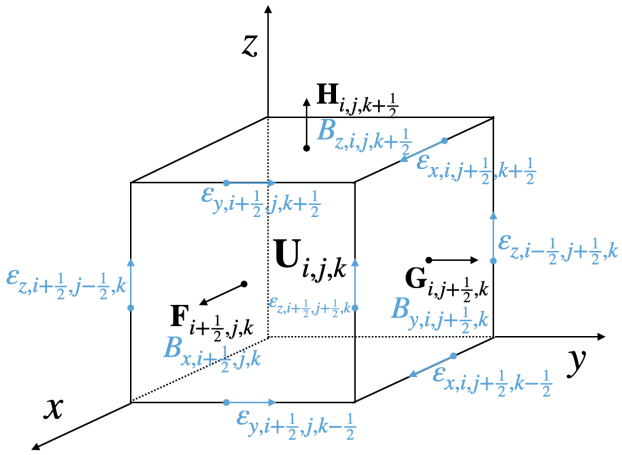

Figure 2: A single cell in a MHD finite-volume simulation. Average values of the conserved variables hydrodynamic variables, ρ, $\vec{v}$, and E, are cell-centered while the magnetic fields, in blue, and fluxes are centered on the faces of the cell. The electric fields, also in blue, are centered on the cell edges.

MHD also requires a new Riemann solver since none of the existing Riemann solvers in Cholla support magnetic fields. Riemann solvers rely on the speed of the various hydrodynamic, or magnetohydrodynamic, waves to select the state that a given interface is occupying. The chosen state determines the exact method used to calculate the flux through that interface. Hydrodynamics has three waves, the left and right moving sound/acoustic waves and the contact discontinuity/entropy wave. MHD in contrast has seven waves as shown in Figure 3. The contact discontinuity remains but the sound waves are split into the fast and slow magnetosonic waves and a new magnetic wave, the Alvén wave, is introduced. These new waves require either extensive modifications to existing Riemann solvers or an entirely new Riemann solver to account for them. We chose to implement the popular HLLD (Harten–Lax–van Leer Discontinuities) Riemann solver which is detailed in the Appendix. This Riemann solver was chosen due to its high level of accuracy and stability across a wide range of parameters. The HLLD Riemann solver is similar in structure to the existing HLLC Riemann solver, and in the presence of zero magnetic fields it mathemati-

cally reduces to the HLLC Riemann solver. Despite the similarities to the HLLC solver the HLLD Riemann solver includes the MHD fast, Alfvén, and entropy waves.. However, like the HLLC Riemann solver, the HLLD Riemann solver is an approximate Riemann solver and one of the primary approximations is that it makes is neglecting the slow magnetosonic wave. Nonetheless, the tests is Section 4 show that it can accurately approximate the slow magnetosonic wave (see Section 4.1.1) and even slow switch-on shocks (see the Ryu & Jones 4D shock tube in Section 4.1.4).

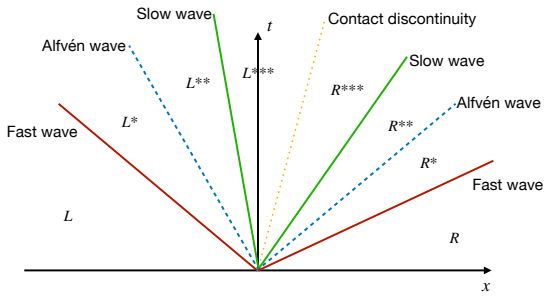

Figure 3: The seven MHD waves with the Riemann solver states labelled.

1.2 Existing Galaxy Simulations

In this section, a brief review of recent MHD galaxy simulations is given. Leveraging state-of-the-art computational techniques and increasingly powerful resources, these simulations illuminate the complex interplay between gas dynamics, magnetic fields, and stellar processes within galaxies. From the formation of galactic magnetic fields to their influence on star formation, galactic outflows, and accretion onto supermassive black holes, MHD simulations provide a framework for understanding the underlying physics driving the observed properties of galaxies across cosmic time. The examples have been chosen to give a broad overview and not to focus on a single subtopic.

Ntormousi et al. 2020 [79] demonstrated the $\alpha-\omega$ dynamo in a global galaxy simulation for the first time. They utilized the RAMSES code which uses constrained transport to

maintain zero divergence. They simulated a $100kpc^3$ with a coarse resolution of 256^3 and six levels of refinement to achieve an effective resolution of 4096^3 ($\approx 24pc$) in the most refined regions. They demonstrated that the $\alpha-\omega$ dynamo is capable of amplifying a magnetic field by a factor of ~ 100 within 500 Myr. They did not include stellar feedback or cosmic rays and their simulations generate magnetic field ~ 100 times weaker than observed magnetic fields.

Van de Voort et al. 2021 [114] studied the overall impact of magnetic fields on the CGM of a $10^{12}M_\odot$ halo mass galaxies. They utilized the moving mesh code AREPO which uses the Powell 8-wave scheme to control magnetic divergence. Since AREPO uses an unstructured Voronoi mesh for the gas a mass per cell rather than specific cell size is targeted for mesh refinement. They targeted a mass of $5.4^4M_\odot$ baryon mass, $2.9^4M_\odot$ dark matter mass, and a maximum volume of $1kpc^3$ within 1.2 virial radii of the center of the simulated galaxy. They did not list the highest effective resolution or refinement levels used. They found that including magnetic fields has no impact on many of the bulk properties such as the stellar masses, ISM masses, and SFR but that including magnetic fields does increase the mass of the AGN, resulting in a more disk dominated galaxy, and making the CGM more filamentary and more collimated (amongst other effects). The major limitations of their simulations are: low resolution and no cosmic rays. If included, both of these could have led to stronger and more impactful magnetic fields.

Farcy et al. 2022 [33] studied the effect of cosmic rays on galaxy dynamics when coupled with stellar radiation and supernovae. They utilized the RAMSES-RT radiation-MHD code which uses constrained transport to maintain zero divergence and the Global Lax-Friedrichs intercel flux function to advect the photon fluid with a reduced speed of light. They used simulation boxes with widths of 150, 300, and 600kpc depending on galaxy mass and maximum resolutions of 9, 9, and 18pc respectively and minimum resolutions of 2.34, 2.34, and 4.68kpc. While not explicitly stated in the paper this works out to coarse resolutions of 64^3, 128^3, and 128^3 respectively with highest effective resolutions of just over $\sim 16k^3$, $\sim 33k^3$, $\sim 33k^3$ and 8 levels of mesh refinement. While their figures show excellent resolution in the disk, the resolution in the CGM is only sufficient for resolving structure on the largest scales. They found that including cosmic rays results in thicker disks with a smoother

ISM, stronger and colder galactic outflows, and that the effects are highly sensitive to the diffusion coefficient used in the cosmic ray model. While they include cosmic rays, radiation, and stellar feedback the resolution in the CGM is too coarse to resolve any dynamics outside of the areas nearest the disk.

1.3 Scientific Software Best Practices

The second major component of this dissertation is the adoption and implementation of several scientific software best practices by the Cholla development team. My work in this area has been primarily concerned with the technical implementation of these best practices when applicable along with spearheading some of the more complex topics such as testing; details can be found in Chapter 5.

As discussed in Section 1.1, software pervades every part of modern astronomy and has been growing ever more critical. This trend is not limited to astronomy but is consistent across most scientific disciplines. Scientists typically spend 30% or more of their time developing software but are primarily self-taught [53, 92]. This often means that they have not been exposed to the tools, techniques, and best practices required for writing high-quality scientific software. While software is often a tool every bit as important as laboratory equipment, scientists often do not know how reliable their software is[56, 55].

To address some of these concerns I, along with the rest of the Cholla development team, have worked to implement many of the recommendations in Wilson et al. 2014 and Wilson et al. 2017[119, 120]. The main best practices that I have worked on implementing are: the integration of a testing framework, automated testing, static analysis, and automated formatting.

1.3.1 Testing Scientific Software

A cornerstone of writing any software, scientific or not, is testing that software to ensure that it operates correctly. In scientific software, this is often done in a manual and very

labor-intensive manner. In a simulation code like Cholla this might look like:

1. Determine a set of test problems/initial conditions with known analytical or numerical solutions.
2. Run each of those problems one-by-one manually.
3. Process the outputs of each problem into a human readable/understandable form through plotting, statistical analysis, etc. This is typically a lossy process that is insensitive to small changes.
4. Compare the human-readable results to the known correct results and determine if they are correct. Depending on the problem, this is often done by eye which is not incredibly precise.

This process is often too time-consuming to be done very often and as such is only done rarely. Consequently, software bugs can often be undetected for extended periods of time. This can directly impact scientific results.

In comparison, a testing framework can run tests in a few minutes, and often completely unattended. Due to this convenience, they can be frequently and easily utilized during the development process where they are the most impactful to the accuracy of the final result. Additionally, a broader, more comprehensive suite of tests can be run. These may include tests that the developer might not have the familiarity or confidence to run manually themselves. Instead of only being able to test the entire code at once (a "system test"), which makes bugs difficult to localize, testing frameworks make it possible to test individual sub-components with reasonable ease. These more focused tests can make it much easier to localize errors, massively speeding up the debugging process.

Tests that can be run rapidly and have their results checked by the computer also enable the tests to be fully automated and run on a schedule or on new code before it is merged into the code base. This helps ensure that the new code is correct and does not have any unintended side effects that can impact other portions of the code.

My work in integrating a testing framework and automated testing into Cholla is described in detail in Chapter 5.

1.3.2 Static Analysis of Scientific Codes

Static analysis is the analysis of the source code of a program without executing that program. It is typically done using automated tools such as Pylint[3] or Clang-Tidy[4]. These tools excel at finding issues in a code base that are easy for humans to make and difficult for them to find. As these tools have evolved they can also be used to check that certain coding standards are met, such as enforcing a specific casing type on variables names which can reduce the cognitive load when reading code[67, 9].

Research has shown that scientific codes contain many issues that are detectable through static analysis[55] and so integrating static analysis tools into scientific code bases is an efficient way to detect and mitigate issues within the code base.

1.3.3 Automatic Code Formatting

In much the same way that consistent variable naming can reduce the cognitive load on the people working on a code base, having standard formatting has similar benefits [67, 9]. People however are not very good at maintaining consistent formatting. Formatting code would require memorizing hundreds of rules and then implementing them perfectly without fail, an impossible task for a human. Tasks like this however are perfect for computers and easy to do automatically with modern software infrastructure. Automated code formatting is easy to implement with significant benefits to developers.

1.4 Summary

Magnetic fields are important in a variety of astrophysical phenomena such as stellar and galactic dynamics [20, 7, 12]. Implementing MHD into Cholla will allow us to create simulations with unprecedented resolution and accuracy. While computationally demanding, grid-based methods of implementing MHD are well-situated to take advantage of the

[3] `https://pypi.org/project/pylint/`
[4] `https://clang.llvm.org/extra/clang-tidy/`

massively parallel nature of GPUs and new exascale supercomputers. At the time of writing Cholla is the only exascale-capable MHD code that utilizes the more accurate constrained transport method for maintaining the zero-divergence condition. The addition of the various scientific software best practices, especially automated testing, allows us to develop and expand Cholla in the future with greater confidence in the correctness and accuracy of the results.

2.0 Introduction

Over the past decade it has become increasingly clear that magnetohydrodynamics (MHD) plays a significant role in a variety of astrophysical phenomena [e.g. 20, 7, 77, 50, 61, 10, 78, 26, 112, 90, 12]. Magnetic fields couple to gas both directly through plasma interactions with the magnetic field, and indirectly through cosmic ray transport [e.g. 89, 42, 19, 16, 117, 122, 43, 80, 116], anisotropic conduction [e.g. 121, 51, 102, 41, 82, 15], and other physical effects.

Many studies have focused on the role of magnetic fields in galaxy evolution and have provided intriguing hints as to the possible effects of magnetic fields in galaxy dynamics [101, 3, 46, 83, 60, 118]. Different simulations employing different numerical methods show varying impacts of magnetic fields, ranging from magnetic fields being largely irrelevant on large scales to magnetic fields being critically important in determining the evolution and structure of the interstellar medium (ISM), modifying galactic feedback, and influencing the structure of the circumgalactic medium (CGM) [72, 29, 65, 52, 84, 4]. One factor in this uncertainty is the effect of numerical resolution on MHD simulations – galactic magnetic fields are likely amplified by a turbulent dynamo, which operates over a large dynamic range [71, 8, 40, 18, 12]. The accuracy with which the dynamo is captured thus depends on both the numerical method that is employed as well as the resolution. Thus, a simple way to extend the dynamic range captured in an MHD simulation is by employing very high resolution MHD simulations – simulations that are now possible thanks to recent developments in hardware, numerical algorithms, and software.

Modern numerical methods for MHD are sophisticated and robust, but even with highly optimized codes, MHD simulations remain very computationally expensive [107]. This computational expense is a result of the high number of floating point calculations required by modern finite-volume methods and the heavy memory bandwidth demands of MHD codes [48]. In addition, MHD turbulent dynamos operate across a large dynamic range, from the full scale of a galaxy all the way down to a few parsecs or smaller, five or more orders of magnitude spatially [87, 88, 79, 36]. As a result, high resolution simulations are critical in

order to accurately capture the effects of magnetic fields in astrophysical simulations.

This need has driven a push to develop MHD codes that can take advantage of modern computer architectures [e.g. 97, 1, 123, 100, 69, 17, 58, 47]. Very large, high resolution MHD simulations require supercomputers to run due to their high computational cost. The primary source of computational power in most new supercomputers is Graphics Processing Units (GPUs)[1]. For example, of the top ten supercomputers named in the November 2023 Top500 list, only one - Fugaku - does not rely on GPUs for the majority of its performance.[2] The ubiquity of GPUs in modern supercomputers thus necessitates the development of GPU-based astrophysical MHD simulation codes.

The induction equation implies a simple constraint; it requires that the magnetic field be divergence free, i.e. the Universe does not contain magnetic monopoles. However, achieving this constraint numerically is not trivial, and several methods have been developed. Among these are the Powell 8 Wave scheme, which adds an additional source term to the induction equation and uses an 8 wave Riemann solver with the 8th wave corresponding to the magnetic divergence [91]; vector potential/projection methods, which project the magnetic field into the scalar and vector potential and then perform a cleaning step to reduce the divergence to zero by solving a Poisson equation [11, 96, 24, 110]; divergence cleaning methods, which operate by modifying the system of conserved equations with corrections that dissipate and propagate out the magnetic divergence [27, 73]; and constrained transport (CT), which evolves face-centered values of the magnetic field and updates them using the electromotive forces [31]. We have chosen to implement the CT method in Cholla due to its overall accuracy, because the algorithm pairs well with static structured grids, and because the computational efficiency of GPUs is a good match to the algebraic complexity of CT.

Constrained transport is formally divergence free, and when implemented numerically it typically results in divergence errors on the order of machine round off error [31, 38, 106, 105, 123, 1]. This is accomplished by tracking magnetic fields on a staggered, face centered grid rather than using cell-centered averages. These face centered values are used in conjunction with Riemann fluxes to calculate edge centered electric fields, and those electric fields are

[1] `https://www.top500.org/lists/top500/2023/11/`

[2] Fugaku employs custom CPUs that utilize vector processors similar to GPUs for much of its performance.

used to update the magnetic field. Thus, the trade-off for a divergence-free method is significant additional algorithmic complexity and associated computational expense, which can make the method more challenging to implement in a particle based scheme or when using unstructured meshes.

The Cholla code (Computational Hydrodynamics On paraLLel Architectures) [98] is a fixed grid, finite volume hydrodynamics code for astrophysics that was designed to run natively on GPU-based supercomputers. It employs an unsplit 3D hydrodynamics integrator based on the Van Leer predictor-corrector method [32, 115] and was designed to be extended to MHD using constrained transport [31]. This work presents the MHD extension of Cholla. Our MHD implementation largely follows the Van Leer + Constrained Transport (VL+CT) method presented in [105] with modifications for GPUs. It also uses an HLLD Riemann solver [75] and includes second [105] and third [34] order reconstruction in the characteristic variables [106]. We also highlight implementation choices that are particularly relevant to solving these equations on GPUs.

The extension of Cholla to include MHD allows the simulation of previously unreachable domains. The VL+CT integrator provides high accuracy results with divergences that are zero to round off error. Given current memory constraints, the code is fast enough that a $\sim 459^3$ cell MHD simulation can be run on a single AMD MI250X Graphics Compute Die (GCD) (logically a single GPU), allowing high resolution simulations to be run with only a small number of local resources. In addition, Cholla scales up to power of the largest available supercomputers, enabling simulations up to $19,278^3 \approx 7.2$ trillion cells on *Frontier*[3], the world's first exascale supercomputer. This will allow MHD simulations of entire galaxies with a constant resolution of a few parsecs per cell, turbulent box simulations with many trillions of cells, or many thousands of lower resolution simulations to be computed rapidly. For example, with approximately the same computing resources used to evolve a $19,278^3$ simulation one could run thousands of $2,000^3$-cell simulations, enabling entire parameter studies with resolutions comparable to current cutting edge CPU-based simulations.

One potential application of Cholla-MHD would be a very high resolution global galaxy simulation. Cholla-MHD enables simulations with resolutions approaching a single parsec

[3] `https://www.olcf.ornl.gov/frontier/`

which could help answer questions about the galactic dynamo [71, 8, 40, 18, 12], dynamics within the CGM [7, 113], turbulence in the CGM [86], and the impact of cosmic rays on acceleration of cold clouds in galactic outflows[62].

In addition to requiring complex algorithms to produce accurate results, modern community-developed simulation codes like Cholla require robust testing and software-development infrastructure to maintain their reliability. This work also presents the implementation of an automated testing/continuous integration (CI) pipeline for Cholla. CI tools have expanded rapidly in functionality and popularity over the last 20 years and their usefulness in scientific software is well established [6, 119, 120]. Particularly in the last 5 years with the advent of GitHub Actions, Jenkins, and similar easily accessible and cheap (or even free) tools, CI pipelines have become much more straightforward to set up and run even for small groups and individuals. We present our implementation of testing and CI that is designed to be straightforward and scalable from a single GPU all the way up to an exascale machine.

The outline of this paper is as follows. In Chapter 3, we describe our implementation of the VL+CT algorithm in detail along with the modifications we made to efficiently run on GPUs. In Chapter 4, we demonstrate the correctness and accuracy of Cholla on a suite of MHD test problems. We also describe Cholla's performance and weak scaling behavior on up to 74,088 GPUs using *Frontier*. In Chapter 5, we discuss the new continuous integration and automated testing framework. We conclude in Chapter 6. Most figure captions end with the phrase "script link" which hyperlinks to the version of the python script which generated that figure. These scripts, and the associated GitHub repository, have sufficient information to reproduce the figures shown in this paper.

3.0 Methods

3.1 Magnetohydrodynamics

Cholla solves the ideal MHD equations in their conserved Eulerian form using a finite volume method [44]. These equations neglect all dissipative processes, including finite viscosity, electrical resistivity, and thermal conductivity. These approximations are reasonable when simulating regions of with very high Reynolds numbers as is common in many astrophysical problems. We note that although we neglect these additional processes at present, the methods implemented here are fully compatible with future extensions to include additional physics that depends on magnetic fields, such as anisotropic conduction, cosmic ray transport, or non-ideal MHD effects.

The ideal MHD equations are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{23}$$

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} - \boldsymbol{B} \otimes \boldsymbol{B} + P_T \boldsymbol{I}) = 0 \tag{24}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + P_T)\boldsymbol{v} + \boldsymbol{B}(\boldsymbol{B} \cdot \boldsymbol{v})) = 0 \tag{25}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} - \nabla \times (\boldsymbol{v} \times \boldsymbol{B}) = 0 \tag{26}$$

where ρ is density, $\boldsymbol{v} = (v_x, v_y, v_z)$ is the velocity vector, t is time, $\boldsymbol{B} = (B_x, B_y, B_z)$ is the magnetic field, $\boldsymbol{I}$ is the identity tensor, $P_T \equiv P + \frac{1}{2}(\boldsymbol{B} \cdot \boldsymbol{B})$ is the total pressure, and E is the total energy per unit volume $E \equiv \epsilon + \frac{1}{2}\rho(\boldsymbol{v} \cdot \boldsymbol{v}) + \frac{1}{2}(\boldsymbol{B} \cdot \boldsymbol{B})$. We adopt units in which the magnetic permeability $\mu_0 = 1$.

Equation 23 describes the conservation of mass, equation 24 describes the conservation of momentum, equation 25 describes the conservation of energy, and equation 26 is the induction equation, which describes the divergence free condition. Cholla uses an ideal gas

equation of state which is $P \equiv \epsilon(\gamma - 1)$ where γ is the adiabatic index and ϵ is the internal energy density.

In practice these equations are used in their vector form, where $\boldsymbol{U}$ and $\boldsymbol{W}$ are the conserved and primitive variables respectively:

$$\boldsymbol{U} = \Big[\rho, \quad \rho v_x, \quad \rho v_y, \quad \rho v_z, \quad E, \quad B_x, \quad B_y, \quad B_z\Big] \tag{27}$$

$$\boldsymbol{W} = \Big[\rho, \quad v_x, \quad v_y, \quad v_z, \quad P, \quad B_x, \quad B_y, \quad B_z,\Big]. \tag{28}$$

The conservation equations, in Cartesian coordinates, can then be rewritten as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} + \frac{\partial \boldsymbol{F_y}}{\partial y} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0 \tag{29}$$

where $\boldsymbol{F_x}$, $\boldsymbol{F_y}$, and $\boldsymbol{F_z}$ are the vectors of fluxes in the x, y, and z direction respectively and are given by

$$\boldsymbol{F_x} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P_T - B_x^2 \\ \rho v_x v_y - B_x B_y \\ \rho v_x v_z - B_x B_z \\ v_x\left(E + p_T\right) - B_x\left(\boldsymbol{v}\cdot\boldsymbol{B}\right) \\ 0 \\ B_y v_x - B_x v_y \\ B_z v_x - B_x v_z \end{bmatrix} \tag{30}$$

$$\boldsymbol{F_y} = \begin{bmatrix} \rho v_y \\ \rho v_y v_x - B_y B_x \\ \rho v_y^2 + P_T - B_y^2 \\ \rho v_y v_z - B_y B_z \\ v_y\left(E + P_T\right) - B_y\left(\boldsymbol{v}\cdot\boldsymbol{B}\right) \\ B_x v_y - B_y v_x \\ 0 \\ B_z v_y - B_y v_z \end{bmatrix} \tag{31}$$

$$\boldsymbol{F_z} = \begin{bmatrix} \rho v_z \\ \rho v_z v_x - B_z B_x \\ \rho v_z v_y - B_z B_y \\ \rho v_z^2 + P_T - B_z^2 \\ v_z \left(E + P_T\right) - B_z \left(\boldsymbol{v} \cdot \boldsymbol{B}\right) \\ B_x v_z - B_z v_x \\ B_y v_z - B_z v_y \\ 0 \end{bmatrix}. \tag{32}$$

3.2 The VL+CT Integrator

To integrate these equations in Cholla, we implement the VL+CT (Van Leer plus Constrained Transport) integrator introduced in [105], along with the HLLD Riemann solver described in [75]. Many of the equations were first described in the context of the constrained-transport extension to the corner-transport upwind (CTU) method [21] described in 2D [38] and 3D [39] for the Athena MHD code [106]. Our specific implementation of the piecewise parabolic method follows [34], which is itself an extension of the original PPM method presented by [22]. The VL+CT integrator is very similar in structure to the MUSCL-Hancock integrator [115, 32, 109], though with significant new additions for Constrained Transport (CT). We note that this is now the default integrator used in Cholla, as we have found it to be more robust than the CTU algorithm described in [98].

Constrained transport treats the magnetic field as a surface averaged quantity centered at cell interfaces rather than a volume averaged, cell centered quantity like the hydro variables; i.e. a staggered grid. Each face stores only the magnetic field perpendicular to that face, i.e. the $x, i+1/2, j, k$ face stores the $B_{x,i+1/2,j,k}$ magnetic field. At each time step, the magnetic field is then updated using edge averaged electric fields computed from the magnetic flux returned by the Riemann solver. Updating the magnetic field with the electric field automatically fulfills the divergence free condition for magnetic fields, assuming that the

initial conditions are also divergence free.

A brief overview of the algorithm for a single time step using the VL+CT integrator is as follows; more detailed discussion of each step is presented in the following subsections.

1. Compute the time step, Δt.
2. Reconstruct interface states using a piecewise constant method (PCM) approximation.
3. Solve the Riemann problem on every interface using the PCM interface states.
4. Compute the edge centered electric fields.
5. Update the $t = n$ conserved variables to $t = n + \Delta t/2$ using the computed fluxes and electric fields.
6. Reconstruct interface states using a piecewise linear method (PLM) or piecewise parabolic method (PPM) approximation.
7. Solve the Riemann problem on every interface using the higher order reconstructed interface states.
8. Recompute the edge centered electric fields.
9. Update the $t = n$ conserved variables to $t = n + \Delta t$ using the half time step fluxes and electric fields

3.2.1 Step 1: Compute the Time Step

The first step is to compute the time step. Cholla implements uniform time steps across the grid, so this is the minimum crossing time of any wave in any cell multiplied by a Courant factor to maintain the Courant–Friedrichs–Lewy condition [23]:

$$\Delta t = C_{CFL} \min \left(\frac{\Delta x}{\mid v^n_{x,i,j,k} \mid + c^n_{f,i,j,k}}, \frac{\Delta y}{\mid v^n_{y,i,j,k} \mid + c^n_{f,i,j,k}}, \frac{\Delta z}{\mid v^n_{z,i,j,k} \mid + c^n_{f,i,j,k}} \right), \tag{33}$$

where Δt is the time step, $C_{CFL} \leq 0.5$ is the CFL number, $\mid v^n_{l,i,j,k} \mid$ is the magnitude of the velocity in the l direction velocity in the i, j, k cell, and c^n_f is the fast magnetosonic wave-speed computed using *cell centered* values. The method is formally accurate at $C_{CFL} = 0.5$, though that is an estimate[105]. We have found empirically that the method is not stable at $C_{CFL} = 0.5$ but is stable at $C_{CFL} = 0.5$[1]. The fast and slow magnetosonic speeds are

[1]Typically we use $C_{CFL} = 0.3$ to improve the accuracy of the operator splitting

$$c_{f,s} = \sqrt{\frac{\gamma p + \mid \boldsymbol{B} \mid^2 \pm \sqrt{\left(\gamma p + \mid \boldsymbol{B} \mid^2\right)^2 - 4\gamma p B_x^2}}{2\rho}} \tag{34}$$

where the + option corresponds to the fast magnetosonic speed, and − to the slow.

Equation 33 computes the minimum crossing time of a wave in a specific cell. A global reduction is then performed to find the minimum in the entire grid; that minimum is used as the time step Δt. This reduction is done primarily on the GPU followed by an MPI ALL_REDUCE. Because GPUs programming methods are inherently parallel, GPU reductions are not trivial and their performance is sensitive to the method used. Details of our reduction method are discussed in section 3.3.

The cell-centered magnetic field is computed with a direct average of the face centered values:

$$\begin{aligned} B^n_{x,i,j,k,} &= \frac{1}{2}\left(B^n_{x,i+1/2,j,k} + B^n_{x,i-1/2,j,k}\right) \\ B^n_{y,i,j,k,} &= \frac{1}{2}\left(B^n_{y,i,j+1/2,k} + B^n_{y,i,j-1/2,k}\right) \\ B^n_{z,i,j,k,} &= \frac{1}{2}\left(B^n_{z,i,j,k+1/2} + B^n_{z,i,j,k-1/2}\right) \end{aligned} \tag{35}$$

These cell-centered values for the magnetic field are used several times throughout the integrator. In CPU-based codes it is typically more efficient to compute these cell centered magnetic fields once and save them, since traditional CPU-based codes are typically compute limited. Cholla is generally limited instead by GPU memory, so we recompute the cell-centered values when they are needed rather than permanently allocating the associated memory.

3.2.2 Step 2: First Order Reconstruction

Reconstructing the interface states at first order, or the piecewise constant method (PCM), is done by setting the primitive interface states values to the same value as the cell:

$$\boldsymbol{W}_{L,i+1/2} = \boldsymbol{W}_{R,i-1/2} = \boldsymbol{W}_i \tag{36}$$

where $\boldsymbol{W}_{L/R,i\pm1/2}$ is the state on the left or right side of the cell. Although Cholla evolves the conserved variables, primitive variables are typically used for interface reconstruction since they are required by the Riemann solver used to compute the fluxes. In higher order spatial reconstructions we also use the characteristic variables derived from the primitive variables.

Although PCM is too diffusive to be used on its own in both reconstruction steps, we note that that mode is excellent for debugging. Here, it offers a computationally-inexpensive way to calculate first-order fluxes, which will be used for the "predictor" step of this predictor-corrector algorithm[115, 105]. Most higher order methods revert to PCM near large discontinuities[68].

3.2.3 Step 3: First Riemann Solve

The next step is to solve the Riemann problem with the first order interface states. To do this, we employ the HLLD Riemann solver introduced in [75]. Because we have implemented the Riemann solver exactly as described in [75] we do not reproduce all the details here. The longitudinal magnetic field does not require reconstruction and can be used directly since it is stored at the face. The transverse fields (i.e. the fields parallel to the interface) are reconstructed from the cell centered average (computed as shown in section 3.2.1), then reconstructed using PCM identically to other fields.

The magnetic fluxes returned by the Riemann solver are the face centered electric fields [see section 5.3 of 106]. For clarity, we list in Table 1 the correlation between flux and electric field.

3.2.4 Step 4: Compute the Constrained Transport Electric Field

The next step is to calculate the constrained transport electric field. These fields are *line averaged* along each cell vertex. These line averaged fields are constructed by averaging the face centered electric fields from the previous step, and their slopes. Equation 37 is used to reconstruct these line averaged electric fields; only the equation for the z-direction is presented, the equations for the other directions can be found by cyclic permutation. The equations for computing the other directions are obtained by substituting out the z index

Magnetic Flux to Face Centered Electric Field.

HLLD Solve Direction	Magnetic Flux	Eqn. as a Cross Product	Electric Field
X	$V_x B_y - B_x V_y$	$(V \times B)_z$	$-\varepsilon_z$
X	$V_x B_z - B_x V_z$	$-(V \times B)_y$	ε_y
Y	$V_x B_y - B_x V_y$	$(V \times B)_z$	$-\varepsilon_x$
Y	$V_x B_z - B_x V_z$	$-(V \times B)_y$	ε_z
Z	$V_x B_y - B_x V_y$	$(V \times B)_z$	$-\varepsilon_y$
Z	$V_x B_z - B_x V_z$	$-(V \times B)_y$	ε_x

Table 1: The directions used here are relative to the internal workings of the HLLD solver. Since the HLLD solver is inherently 1D we run it once for each of the faces of a cell. So in the case where the solver is running in the Y direction the solver's Y field is actually the Z field and the solver's Z field is actually the X field, cyclically extended for the Z direction.

with x or y and changing the derivatives appropriately.

$$\begin{aligned}\mathcal{E}_{z,i-1/2,j-1/2,k} = \frac{1}{4}\left(\mathcal{E}_{z,i-1/2,j,k} + \mathcal{E}_{z,i,j-1/2,k} + \mathcal{E}_{z,i-1/2,j-1,k} + \mathcal{E}_{z,i-1,j-1/2,k}\right) \\ + \frac{\Delta y}{8}\left(\left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1/2,j-1/4,k} + \left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1/2,j-3/4,k}\right) \\ + \frac{\Delta x}{8}\left(\left(\frac{\partial \mathcal{E}_z}{\partial x}\right)_{i-1/4,j-1/2,k} + \left(\frac{\partial \mathcal{E}_z}{\partial x}\right)_{i-3/4,j-1/2,k}\right)\end{aligned} \tag{37}$$

$\mathcal{E}_{z,i-1/2,j-1/2,k}$ is the line averaged electric field; the first four terms on the right are the face averaged electric fields, and the four derivative terms are the derivatives of those fields in the direction towards the edge, details of which are in Equation 39. Figure 4 shows the spatial relationships between the derivatives and the relevant edge states and reference fields. Each edge requires 4 derivatives and they are computed as differences between a reference state and an edge state.

Technically, constrained transport uses the magnetic flux as the conserved variable and EMF (ElectroMotive Force) to update it. However, because those values only differ from the magnetic flux density (i.e. the magnetic field) and the electric field respectively by factors of unit length either can be treated as the conserved variable, the same way either density or mass can be evolved in the hydro fields [106]. As such we use the magnetic field and electric field to evolve the grid. Electric fields have the proper units to evolve the magnetic flux density, B, whereas EMF has the proper units to evolve the magnetic flux.

On any face there are two non-zero electric fields; both transverse to the face. The component that is used to calculate the field along a given edge is the component that is parallel to that edge; i.e. if the edge points along the z-direction then the field pointing along the z-direction is used, not the field in the x or y direction.

The derivatives from Equation 37 are computed using the upwinded slope as follows

$$\left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1/2,j-1/4,k} = \begin{cases} \left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1,j-1/4,k} & \text{for } v_{x,i-1/2} > 0 \\ \left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i,j-1/4,k} & \text{for } v_{x,i-1/2} < 0 \\ \frac{1}{2}\left(\left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i-1,j-1/4,k} + \left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i,j-1/4,k}\right) & \text{otherwise} \end{cases} \tag{38}$$

where, for example, the derivatives are given by

$$\left(\frac{\partial \mathcal{E}_z}{\partial y}\right)_{i,j-1/4,k} = 2\left(\frac{\mathcal{E}_{z,i,j-1/2,k} - \mathcal{E}^{ref}_{z,i,j,k}}{\Delta y}\right). \tag{39}$$

$\mathcal{E}^{ref}_{z,i,j,k}$, is the cell centered reference field. This reference field is computed with the following cross product

$$\mathcal{E}^{ref,n}_{i,j,k} = -\left(\boldsymbol{v}^n_{i.j.k} \times \boldsymbol{B}^n_{i.j.k}\right). \tag{40}$$

3.2.5 Step 5. Perform the Half Time-step Update

We first update the density, momenta, and energy, but not the magnetic fields, using the standard conservative update equation and the first-order fluxes from Step 3:

$$\begin{aligned} \boldsymbol{U}^{n+1/2}_{i,j,k} = \boldsymbol{U}^n_{i,j,k} &- \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^n_{x,i+1/2,j,k} - \boldsymbol{F}^n_{x,i-1/2,j,k}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\boldsymbol{F}^n_{y,i,j+1/2,k} - \boldsymbol{F}^n_{y,i,j+1/2,k}\right) \\ &- \frac{\Delta t}{\Delta z}\left(\boldsymbol{F}^n_{z,i,j,k+1/2} - \boldsymbol{F}^n_{z,i,j,k+1/2}\right). \end{aligned} \tag{41}$$

We then update the magnetic field using the electric fields computed in Step 4:

$$\begin{aligned} B^{n+1/2}_{x,i-1/2,j,k} = B^n_{x,i-1/2,j,k} &+ \frac{\Delta t}{\Delta z}\left(\mathcal{E}^n_{y,i-1/2,j,k+1/2} - \mathcal{E}^n_{y,i-1/2,j,k-1/2}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\mathcal{E}^n_{z,i-1/2,j+1/2,k} - \mathcal{E}^n_{z,i-1/2,j-1/2,k}\right) \end{aligned} \tag{42}$$

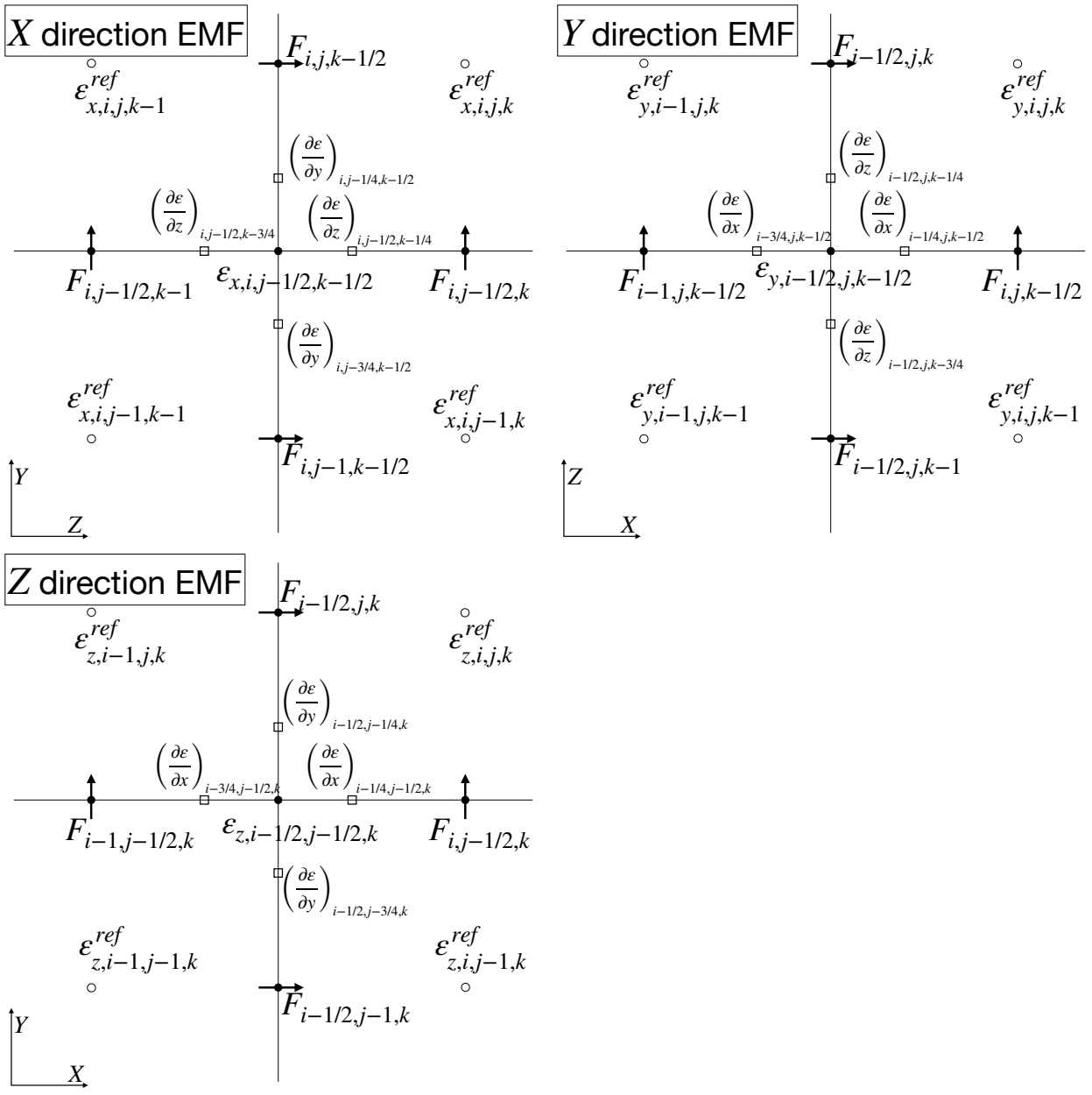

Figure 4: 2D slices in all three planes showing the location of the fluxes, edge electric fields, and derivatives. Based on Figure 5 of [106].

$$B^{n+1/2}_{y,i,j-1/2,k} = B^{n}_{y,i,j-1/2,k} + \frac{\Delta t}{\Delta x}\left(\mathcal{E}^{n}_{z,i+1/2,j-1/2,k} - \mathcal{E}^{n}_{z,i-1/2,j-1/2,k}\right) - \frac{\Delta t}{\Delta z}\left(\mathcal{E}^{n}_{x,i,j-1/2,k+1/2} - \mathcal{E}^{n}_{x,i,j-1/2,k-1/2}\right) \tag{43}$$

$$B^{n+1/2}_{z,i,j,k-1/2} = B^{n}_{z,i-1/2,j,k} + \frac{\Delta t}{\Delta y}\left(\mathcal{E}^{n}_{x,i,j+1/2,k-1/2} - \mathcal{E}^{n}_{x,i,j-1/2,k-1/2}\right) - \frac{\Delta t}{\Delta x}\left(\mathcal{E}^{n}_{y,i+1/2,j,k-1/2} - \mathcal{E}^{n}_{y,i-1/2,j,k-1/2}\right). \tag{44}$$

3.2.6 Step 6. Half Time-step Second Order Reconstruction

Step 5 results in first order time-averaged values for the cell-centered conserved variables and face-centered magnetic fields. To make the integration second-order in time, we need to perform a "corrector" step. First, we perform a higher order interface reconstruction. The method shown here is for Piecewise Linear Method (PLM), reconstruction. Cholla currently implements piecewise constant, piecewise linear, and piecewise parabolic reconstruction, with limiting in the characteristic variables, for MHD. The piecewise parabolic method that Cholla utilizes is discussed in detail in [34]. Using the third order piecewise parabolic method for spatial reconstruction does typically give slightly more accurate results at a given resolution compared to PLM, but since the method is formally second order it does not improve the overall order of convergence. Note that at a given face only the transverse components of the electric field need to be reconstructed. The longitudinal component is already given at the face.

The steps of the PLM update are:

1. Compute the primitive variables from the conserved variables.
2. Compute the left, right, centered, and Van Leer differences in the primitive variables

$$\delta \boldsymbol{W}_{L,i} = \boldsymbol{W_i} - \boldsymbol{W_{i-1}} \tag{45}$$

$$\delta \boldsymbol{W}_{R,i} = \boldsymbol{W_{i+1}} - \boldsymbol{W_i} \tag{46}$$

$$\delta \boldsymbol{W}_{C,i} = \frac{\boldsymbol{W_{i+1}} - \boldsymbol{W_{i-1}}}{2} \tag{47}$$

$$\delta \boldsymbol{W}_{VL,i} = \begin{cases} \frac{2\boldsymbol{W}_{L,i}\boldsymbol{W}_{R,i}}{\boldsymbol{W}_{L,i}+\boldsymbol{W}_{R,i}}, & \text{if } \boldsymbol{W}_{L,i}\boldsymbol{W}_{R,i} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{48}$$

3. Project the slopes into the characteristic variables, $\boldsymbol{a}$, using the eigenvectors listed in the appendix of [106]. Note that to maintain mathematical consistency we use the

eigenvectors of the $\boldsymbol{w_i}$ cell for all four slopes and the later projection back to primitive variables.

4. Apply monotonicity constraints to the characteristic differences to ensure that the reconstruction is total variation diminishing (TVD). We use the following limiter, given in [68]:

$$\delta \boldsymbol{a}_{m,i} = \begin{cases} \text{sgn}(\boldsymbol{a}_{C,i}) \min(2|\boldsymbol{a}_{L,i}|, 2|\boldsymbol{a}_{R,i}|, |\boldsymbol{a}_{C,i}|, |\boldsymbol{a}_{VL,i}|), & \text{if } \boldsymbol{a}_{L,i}\boldsymbol{a}_{R,i} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{49}$$

5. Project the limited characteristic slopes, $\delta \boldsymbol{a}_m$, back into the primitive variables, $\delta \boldsymbol{W}_m$, using the eigenvectors.
6. Compute the interface states $\boldsymbol{W}_{L,i+1/2}$ and $\boldsymbol{W}_{R,i-1/2}$:

$$\begin{aligned} \boldsymbol{W}_{L,i+1/2} &= \boldsymbol{W}_i + \frac{\delta \boldsymbol{W}_{m,i}}{2} \\ \boldsymbol{W}_{R,i-1/2} &= \boldsymbol{W}_i - \frac{\delta \boldsymbol{W}_{m,i}}{2} \end{aligned} \tag{50}$$

where $\boldsymbol{W}_{L/R,i\pm1/2}$ is the state on the left or right side of the cell and $\delta \boldsymbol{W}_{m,i}$ is the monotonically limited primitive slope. Limiting can be done in the primitive variables by skipping steps 3 and 5 and replacing the characteristic slopes in Equation 49 with their primitive counterparts.

3.2.7 Step 7. Second Riemann Solve

Solve the Riemann problem for each interface again using the higher order interface states computed in step 6.

3.2.8 Step 8. Compute the Constrained Transport Electric Fields

Repeat step 3, but using the fluxes from the second Riemann solve and the half time step MHD variables computed in Step 5.

3.2.9 Step 9. Perform the Full Time-step Update

Update the cell-centered hydro variables from $t = n$ to $t = n + \Delta t$ using the conserved update equation and the second order fluxes computed in Step 7:

$$\begin{aligned} \boldsymbol{U}^{n+1}_{i,j,k} = \boldsymbol{U}^{n}_{i,j,k} &- \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^{n+1/2}_{x,i+1/2,j,k} - \boldsymbol{F}^{n+1/2}_{x,i-1/2,j,k}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\boldsymbol{F}^{n+1/2}_{y,i,j+1/2,k} - \boldsymbol{F}^{n+1/2}_{y,i,j+1/2,k}\right) \\ &- \frac{\Delta t}{\Delta z}\left(\boldsymbol{F}^{n+1/2}_{z,i,j,k+1/2} - \boldsymbol{F}^{n+1/2}_{z,i,j,k+1/2}\right). \end{aligned} \tag{51}$$

Update the face-centered magnetic field using the electric fields calculated in Step 8:

$$\begin{aligned} B^{n+1}_{x,i-1/2,j,k} = B^{n}_{x,i-1/2,j,k} &+ \frac{\Delta t}{\Delta z}\left(\mathcal{E}^{n+1/2}_{y,i-1/2,j,k+1/2} - \mathcal{E}^{n+1/2}_{y,i-1/2,j,k-1/2}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\mathcal{E}^{n+1/2}_{z,i-1/2,j+1/2,k} - \mathcal{E}^{n+1/2}_{z,i-1/2,j-1/2,k}\right) \end{aligned} \tag{52}$$

$$\begin{aligned} B^{n+1}_{y,i,j-1/2,k} = B^{n}_{y,i,j-1/2,k} &+ \frac{\Delta t}{\Delta x}\left(\mathcal{E}^{n+1/2}_{z,i+1/2,j-1/2,k} - \mathcal{E}^{n+1/2}_{z,i-1/2,j-1/2,k}\right) \\ &- \frac{\Delta t}{\Delta z}\left(\mathcal{E}^{n+1/2}_{x,i,j-1/2,k+1/2} - \mathcal{E}^{n+1/2}_{x,i,j-1/2,k-1/2}\right) \end{aligned} \tag{53}$$

$$\begin{aligned} B^{n+1}_{z,i-1/2,j,k} = B^{n}_{z,i-1/2,j,k} &+ \frac{\Delta t}{\Delta y}\left(\mathcal{E}^{n+1/2}_{x,i,j+1/2,k-1/2} - \mathcal{E}^{n+1/2}_{x,i,j-1/2,k-1/2}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\mathcal{E}^{n+1/2}_{y,i+1/2,j,k-1/2} - \mathcal{E}^{n+1/2}_{y,i-1/2,j,k-1/2}\right). \end{aligned} \tag{54}$$

3.2.10 Step 10. Increment the Time by Δt

Increment the time by Δt. Additional physics modules are added in an operator-split fashion after this point (chemistry, radiation transport, etc.), after which all MPI communication is done to exchange the ghost/halo cells that surround each MPI rank's subdomain.

3.3 Implementation on GPUs

3.3.1 Memory bandwidth constraints

While the implementation of MHD on GPUs is similar to the implementation on CPUs, there are some crucial differences, especially regarding data handling and movement. Compared to CPUs, GPUs have higher memory bandwidth and extremely high FLOPS, but limited memory capacity and limited functionality due to their fundamentally SIMD (Single Instruction Multiple Data) nature. Also, while GPU memory bandwidth is higher on the whole, due to their parallel nature GPUs can request many more values from memory at once, meaning GPU-based codes are often still memory bandwidth bound.

This leads to some implementation choices that may seem counter-intuitive. For example, while an optimized CPU-based code may compute the cell-centered magnetic fields once and then save them to memory, Cholla recomputes them in each function call where they are required. Similarly, Cholla does not store the primitive variables, only the conserved ones, and recomputes the primitive variables as needed. This approach reduces global memory usage by not storing an entire second grid. It also generally reduces memory bandwidth requirements, since often both the conserved and primitive variables are needed within a function, but only one set needs to be loaded.

Another potential source of memory bandwidth optimization is fusing functions together. By combining multiple GPU kernels into one, different functions can share data without a write/read cycle to global memory. For example, we have implemented this in Cholla by combining the first order reconstruction and Riemann solver kernels, since each interface that is reconstructed is only used in the next Riemann solve, and does not need to be used later in the integrator. Fusing the PCM "reconstruction" into the Riemann solver led to a $\sim 10\%$ improvement in overall runtime. However, fusing the PLMC reconstruction into the Riemann solver actually slowed the code down by $\sim 10 - 15\%$. This slowdown is caused by the increased register usage of the new, larger kernel, which in turn reduced overall GPU occupancy. Thus, the trade-off between reducing global memory accesses and increasing register usage is not always straightforward to determine, and we have found that it depends

on compiler optimizations which can vary between platforms, as well.

Another major challenge in GPU computing is CPU-to-GPU bandwidth which is much lower than GPU memory bandwidth. To address this problem, Cholla now keeps as much data as possible in the GPU memory instead of moving it to and from main system memory, in contrast with previous versions of the code. Although it has always been a GPU-native code, historically Cholla used an extreme version of the "offload" model of GPU programming, keeping the "primary" data (like conserved variable arrays) in CPU memory, and copying that data to the GPU to carry out computations before copying it back each time step. Although this approach originally had the advantage of allowing larger grid sizes to be computed on a single GPU when GPU memory sizes were extremely limited, as GPUs have gotten dramatically faster and their memory capacity has increased over the last decade, these full-grid copies between CPU and GPU memory would now dominate the simulation run time. In addition, most new supercomputers, such as Frontier, have GPUs that are directly connected to the network, so direct off-node GPU-to-GPU MPI communication is now possible. Between these two factors it is much more efficient to store all the simulation data on the GPU, and only move it back to the CPU for i/o operations. In the new version of Cholla described in this work, all computations are carried out on the GPU, and MPI communication proceeds directly from GPU memory buffers. The only functions that remain on the CPU are the setting of the initial conditions, which only occurs once, and reading and writing output files.At the time of writing, the HDF5 library does not support writing files directly from GPU memory.

3.3.2 Performance portability

Cholla is written in C++ and CUDA, the C++ extension developed by NVIDIA. However, CUDA code can only be compiled using NVIDIA's compiler `nvcc` for NVIDIA GPUs. Thus, with the advent of AMD-based supercomputers like *Frontier*, the problem of code portability becomes relevant. We approached this problem in two ways. First, AMD has developed HIP, a cross platform equivalent to CUDA, which can be compiled using their `hipcc` to target either NVIDIA or AMD hardware. Although porting Cholla's CUDA code

to HIP did not prove challenging due to the extreme similarity between the two platforms, rewriting the entire code base in HIP has the drawback of making it impossible to compile on systems with NVIDIA GPUs that do not have HIP installed. Thus, we chose to employ a less invasive modification: we introduced a header file (`gpu.hpp`) which uses C++ preprocessor macros to convert between CUDA and HIP versions of functions at compile time. When building the code, the user merely has to specify whether the compilation should use `hipcc` or `nvcc`, and the code is compiled into HIP or CUDA accordingly. This method also has the advantage of continuing to allow us to maintain a single code-base written in CUDA.

3.3.3 GPU reductions

There are several places in Cholla where a grid wide reduction must be performed. The primary example is the calculation of the time step, which requires finding the minimum crossing time in the full simulation grid. While CPU and MPI based reductions are relatively simple to implement, often through library calls, the inherently parallel nature of the GPU programming model makes GPU based reductions somewhat more complex. For GPU reductions in Cholla, we use a method similar to that described by NVIDIA[2] which describe the challenges of a GPU reduction well. This reduction method uses atomics for the final level of reduction. While this method is straightforward to implement in HIP, at the time of writing, CUDA does not support floating point `atomicMax`. To work around this we have adopted a method from the RAPIDS cuML library[3] which, with some encoding, uses the integral `atomicMax`. Overall this method performs slightly faster than Cholla's previous hybrid GPU+CPU reduction for the number of elements we typically encounter. Primarily though, it is much simpler to use and performs dramatically better as the number of elements increases.

[2] `https://developer.nvidia.com/blog/faster-parallel-reductions-kepler/`

[3] `https://github.com/rapidsai/cuml/blob/dc14361ba11c41f7a4e1e6a3625bbadd0f52daf7/cpp/src_prims/stats/minmax.cuh`

4.0 MHD Tests

In this Chapter, we show the results of a suite of test problems that demonstrate the accuracy, robustness and performance of Cholla MHD. We have included many problems that are common in the literature and specifically selected tests that are challenging for the integrator and, when possible, have quantifiable measures of correctness for comparison to other methods and codes. All of these problems are also included in our automated test suite as either accuracy and/or regression system tests (see Chapter 5 for details). Section 4.1 discusses tests for accuracy, while Section 4.2 discusses the performance and scaling of Cholla MHD.

4.1 Accuracy Tests

4.1.1 Linear Wave Convergence

The propagation of the four MHD linear waves provides an excellent quantitative measure of the accuracy of a numerical MHD method. Our linear wave tests use a periodic domain of $1.0 \times 1.0 \times 1.0$ and a resolution of $N \times 16 \times 16$ where N goes from 16 to 512 in powers of 2. The wave equation is

$$q = \overline{q} + AR_w \sin \frac{2\pi x}{\lambda}, \tag{55}$$

where q is the conserved variable, $\overline{q}$ is the mean background state, $A = 10^{-6}$ is the amplitude of the wave, R_w is the right eigenvector in conserved variables for the wave mode w, x is the position, and $\lambda = 1$ is the wavelength of the wave. The adiabatic index γ is 5/3 and the background state is: $\overline{\rho} = 1.0$, $\overline{v_x} = \overline{v_y} = \overline{v_z} = 0$ (except for the contact wave where $\overline{v_x} = 1$), $\overline{P} = 1/\gamma$, $\overline{B_x} = 1$, $\overline{B_y} = 1.5$, and $\overline{B_z} = 0$. The right eigenvectors for this state are given in Appendix A of [39].

The wave propagates for one period, after which the error is computed between the

initial and final state. First we compute the L1 norm, which is the absolute difference for each conserved variable between the initial and final state:

$$\delta q_s = \frac{1}{n_x n_y n_z} \sum_{i,j,k} \mid q^f_{i,j,k,s} - q^i_{i,j,k,s} \mid, \tag{56}$$

where q_s is a specific conserved variable. We then compute the L2 norm of this vector of L1 norms as

$$\mid\mid \delta q \mid\mid = \sqrt{\sum_s (\delta q_s)^2}. \tag{57}$$

These L2 errors are plotted in Figure 5 for both the PLM and PPM reconstructions. The results are comparable to the results in [105] and demonstrate the expected second order convergence. Using PPM improves the accuracy of the solution by approximately an order of magnitude at any given resolution, but maintains the second order convergence due to the second order nature of the integrator. We have implemented these tests in all three directions with the waves moving in the positive or negative directions and find identical results.

4.1.2 Circularly Polarized Alfvén Wave

The circularly polarized Alfvén wave is a non-linear wave that tests a code's accuracy in the non-linear regime with the quantitative benefits of a regular wave test [111]. The tests use a periodic domain of $3.0 \times 1.5 \times 1.5$ and a resolution of $2N \times N \times N$ where N goes from 8 to 256 in powers of 2. The wave is initialized at an oblique angle the grid, making this a fully 3D test.

In a coordinate system aligned with the movement of the wave, the initial conditions are $\rho = 1.0$, $P = 0.1$, $v_x = (0, -1)$ for traveling or standing waves respectively, $v_y = A \sin \frac{2\pi x}{\lambda}$, $v_z = A \cos \frac{2\pi x}{\lambda}$, $B_x = 1.0$, $B_y = A \sin \frac{2\pi x}{\lambda}$, and $B_z = A \cos \frac{2\pi x}{\lambda}$, where the amplitude of the wave $A = 0.1$ and the wavelength $\lambda = 1.0$. These coordinates are then transformed with the rotation

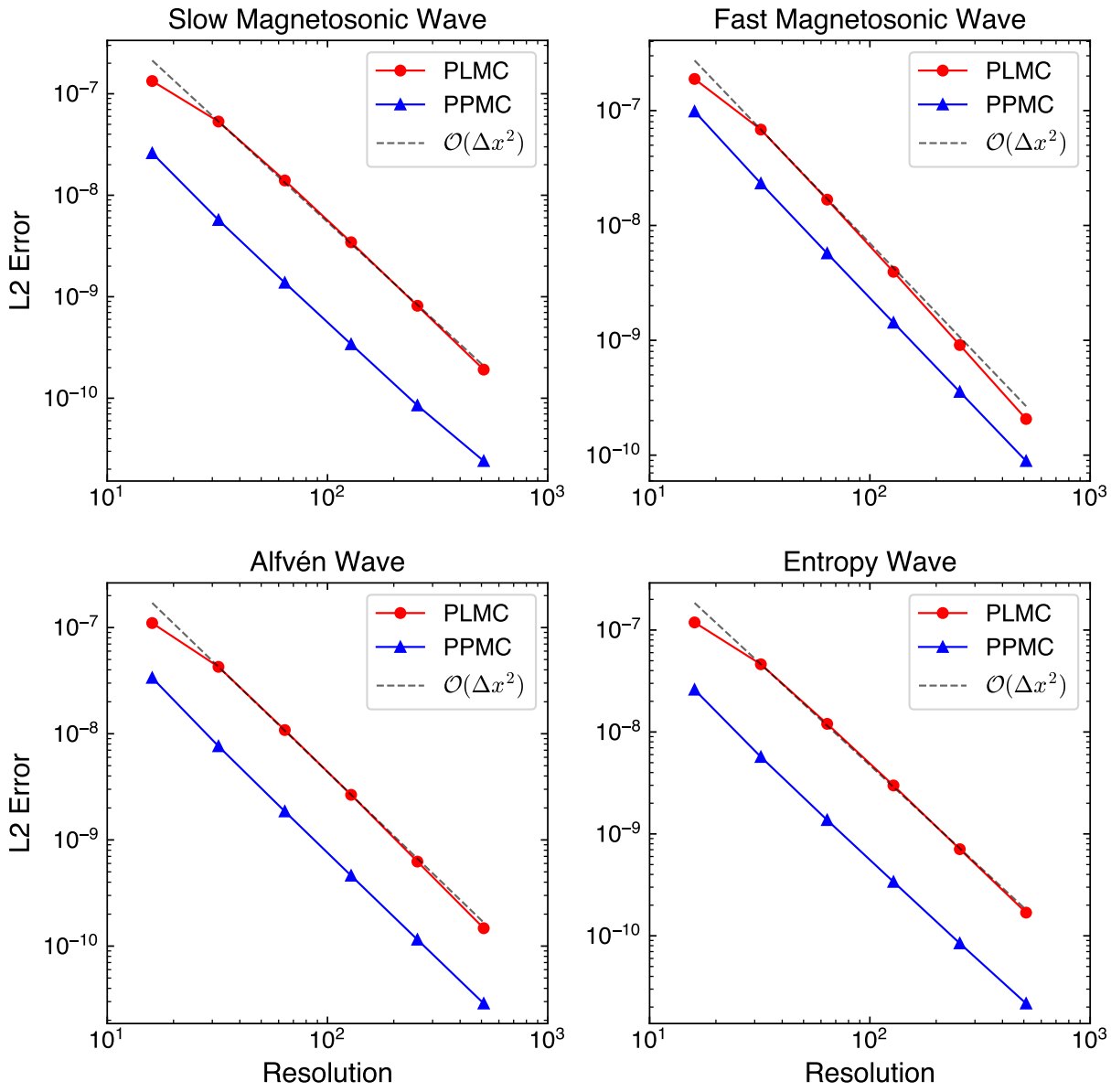

Figure 5: Linear Wave Convergence of all four MHD waves using PLM and PPM reconstruction. script link

$$x\prime = x\cos\alpha\cos\beta - y\sin\beta - z\sin\alpha\cos\beta$$

$$y\prime = x\cos\alpha\sin\beta + y\cos\beta - z\sin\alpha\sin\beta$$

$$z\prime = x\sin\alpha + z\cos\alpha$$

with $\sin\alpha = 2/3$ and $\sin\beta = 1/\sqrt{5}$. This ensures the domain is fully periodic through the boundaries and the wave can travel (or stand) indefinitely. The magnetic fields are initialized with the vector potential to ensure initial divergence is zero to round off. The waves are then run for a single period and the L2 norm of the L1 error vector is plotted in Figure 6 using the same method as in Section 4.1.1. It is interesting to note that the accuracy improvement of PPM versus PLM seen for the linear waves is absent in this non-linear test.

These Alfvén waves are subject to a parametric instability [28], which should not be present for these initial conditions. However, the truncation error will result in small variations in the magnetic pressure which drives low amplitude compression waves [106].

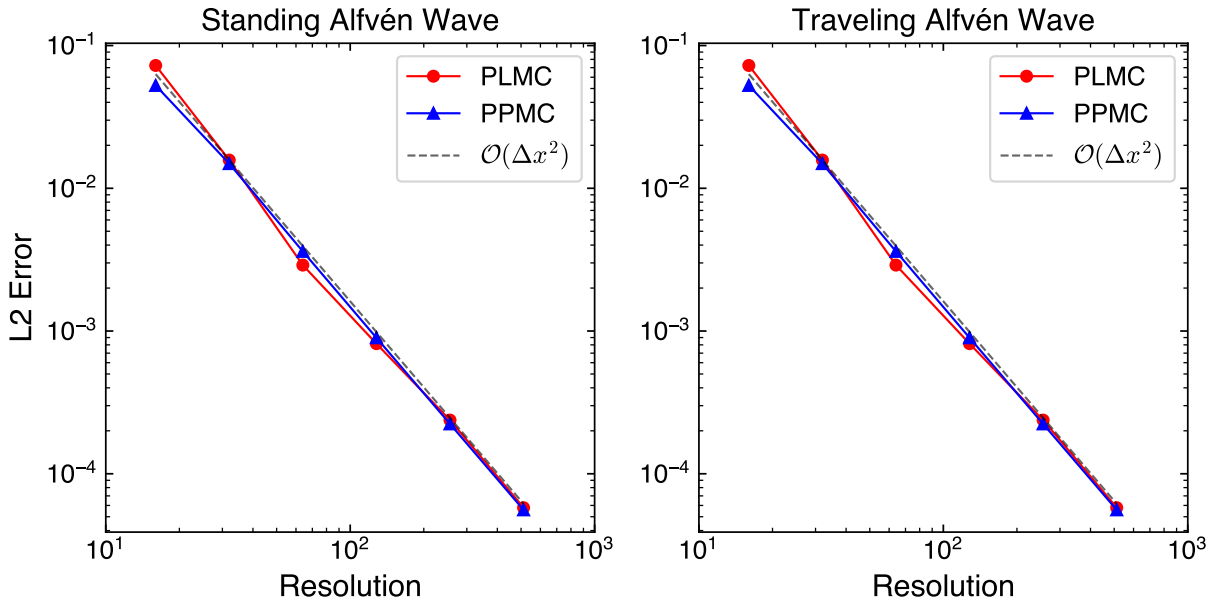

Figure 6: Circularly Polarized Alfvén Wave Convergence using PLM and PPM reconstruction. script link

4.1.3 Advecting Field Loop

The advecting field loop test consists of a tilted spherical current loop which travels across the domain at an oblique angle to the grid. This test requires particularly accurate balancing of the non-zero components of the induction equation. It also has zero magnetic

field outside the spherical current loop; as the current loops moves across the grid those cells that are no longer in the loop should return to zero to within round off error. It is also a good test of the dissipation of the magnetic field, as the magnetic pressure should remain constant.

The initial conditions for this test are most easily described using the magnetic vector potential, which is the vector quantity whose curl equals the magnetic field, $\boldsymbol{B} = \nabla \times \boldsymbol{A}$. The background state is $\rho = 1.0$, $P = 1.0$, $v_x = 1.0$, $v_y = 1.0$, $v_z = 2.0$, $B_x = 0$, $B_y = 0$, and $B_z = 0$.

In the central region the state is given by the following vector potential, which we have chosen such that $A_x = 0$:

$$A_y = A_z = \begin{cases} A\left(R - r\right), & \text{for } r < R \\ 0, & \text{otherwise} \end{cases} \tag{58}$$

where r is the Euclidean distance from the center of the domain, $R = 0.3$, and the amplitude $A = 10^{-3}$. Note that since the vector potential is along the vertices of the cells, A_y and A_z will never have the same value at the same position as they are not stored at identical positions. The test is conducted on a grid of $N \times N \times 2N$ cells for $N = (32, 64, 128, 256)$ with a periodic domain of $1.0 \times 1.0 \times 2.0$ centered at zero and evolved for two periods; $t_{max} = 2.0$.

Figure 7 shows the mean of cell centered B^2, normalized to the initial value, in order to demonstrate the convergence of the dissipation rate. The dissipation rate is comparable to those found in the literature [106] and improves at approximately first order. Figure 7 also shows the maximum divergence in the domain as a function of time. Throughout the entire evolution it remains near round off and, after an initial rise, remains fairly constant. The zero magnetic field region outside of the current loop also remains near zero throughout the entire evolution of the problem. Figure 8 shows cross sections of the loop initial conditions and after one period with a resolution of $128 \times 128 \times 256$ cells. The shape is well maintained with minimal dissipation.

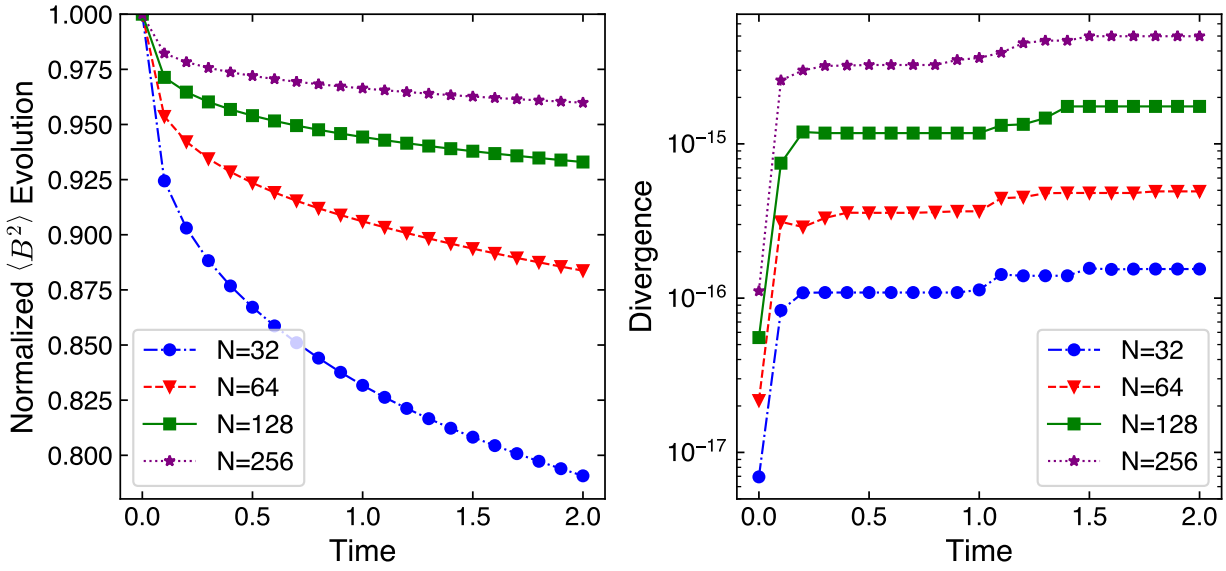

Figure 7: Evolution of tilted spherical magnetic field loop through two full periods using PPMC reconstruction. Mean of B^2 normalized to the initial value as a function of time (left) and the maximum divergence in the domain as a function of time (right). script link

4.1.4 MHD Riemann Problems

The typical Riemann problem setup uses a domain with a discontinuity at the midpoint between two different states. As the problem evolves in time, the waves propagate along characteristics such that the solution evolves self-similarly in time. All of Riemann problems shown in this section employ a domain of $1 \times 1 \times 1$ and resolution of $512 \times 16 \times 16$ and are run until the t_{max} which is specified for that particular problem. We use parabolic reconstruction with limiting in the characteristic variables unless otherwise noted. All have been run in all three spatial directions with both possible orientations of the two states and achieved identical results. The details of each left and right state are given in Table 2.

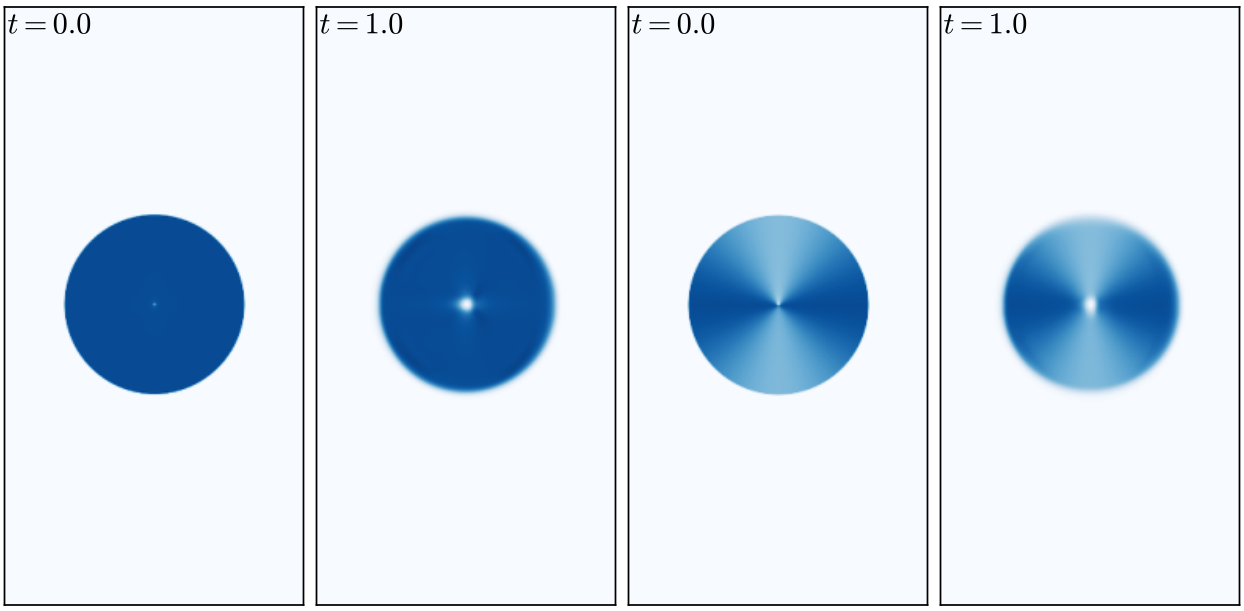

Figure 8: Cross sections of the spherical advecting field loop magnetic energy density at $t = 0.0$ and one period. The first and second panels show a slice centered on the loop through the plane of symmetry. The third and fourth panels show a slice along the $x - z$ plane. Note that these figures utilize PLMC reconstruction as PPMC introduced spurious oscilations in the direction of advection. script link

4.1.4.1 Brio & Wu Shock Tube

Figure 9 shows the Brio & Wu Shock Tube [13] which is a staple test of MHD codes. This Riemann problem is essentially the Sod shock tube [103] with a magnetic field. However, this shock tube is an excellent stress test for PPM reconstruction, as methods higher than second order tend to create large oscillations in the solution due to the slowly moving shock waves. We have implemented both the PPM reconstruction algorithm used in the VL+CT integrator from the method described in [106] as well as the method described in [34], and find that the latter significantly reduces oscillations for this test. With PPM reconstruction, we find oscillations in the solution when limiting in either the primitive or characteristic variables. No oscillations are present when using PLM reconstruction.

Riemann Problem Initial Conditions.

Riemann Problem	γ	t_{max}	B_x	ρ_L	P_L	$v_{x,L}$	$v_{y,L}$	$v_{z,L}$	$B_{y,L}$	$B_{z,L}$
Brio & Wu	2	0.1	0.75	1	1	0	0	0	1	0
Dai & Woodward	$\frac{5}{3}$	0.2	$\frac{2}{\sqrt{4\pi}}$	1.08	0.95	1.2	0.01	0.5	$\frac{3.6}{\sqrt{4\pi}}$	$\frac{2}{\sqrt{4\pi}}$
Ryu & Jones 1a	$\frac{5}{3}$	0.08	$\frac{5}{\sqrt{4\pi}}$	1	20	10	0	0	$\frac{5}{\sqrt{4\pi}}$	0
Ryu & Jones 4d	$\frac{5}{3}$	0.16	0.7	1	1	0	0	0	0	0
Einfeldt Rarefaction	1.4	0.16	0	1	0.45	-2	0	0	0.5	0

Riemann Problem	ρ_R	P_R	$v_{x,R}$	$v_{y,R}$	$v_{z,R}$	$B_{y,R}$	$B_{z,R}$
Brio & Wu	0.125	0.1	0	0	0	-1	0
Dai & Woodward	1	1	0	0	0	$\frac{4}{\sqrt{4\pi}}$	$\frac{2}{\sqrt{4\pi}}$
Ryu & Jones 1a	1	1	-10	0	0	$\frac{5}{\sqrt{4\pi}}$	0
Ryu & Jones 4d	0.3	0.2	0	0	1	1	0
Einfeldt Rarefaction	1	0.45	2	0	0	0.5	0

Table 2: The L/R subscripts indicate that it is the left/right state. B_x is always the same in both states.

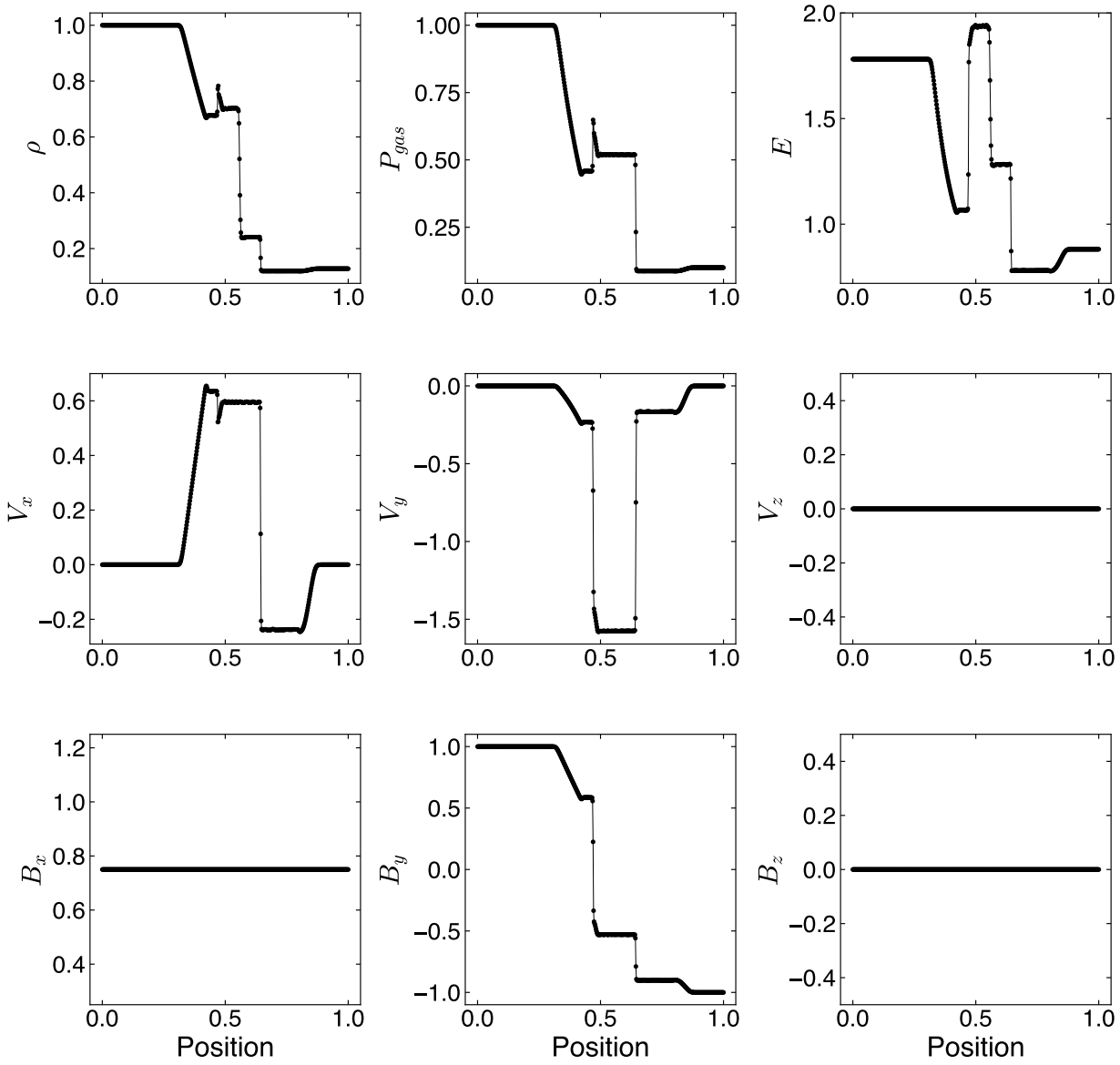

Figure 9: The Brio & Wu Shock Tube solution [13]. script link

4.1.4.2 Dai & Woodward Shock Tube

Figure 10 shows the Dai & Woodward Shock Tube (also called Ryu & Jones 2a) [25, 95] which produces all seven possible MHD waves. From left to right they are: fast shock, Alfvén

wave, slow shock, contact discontinuity, slow shock, Alfvén wave, and fast shock. This makes it an excellent laboratory for checking that the full spread of wave modes are well resolved.

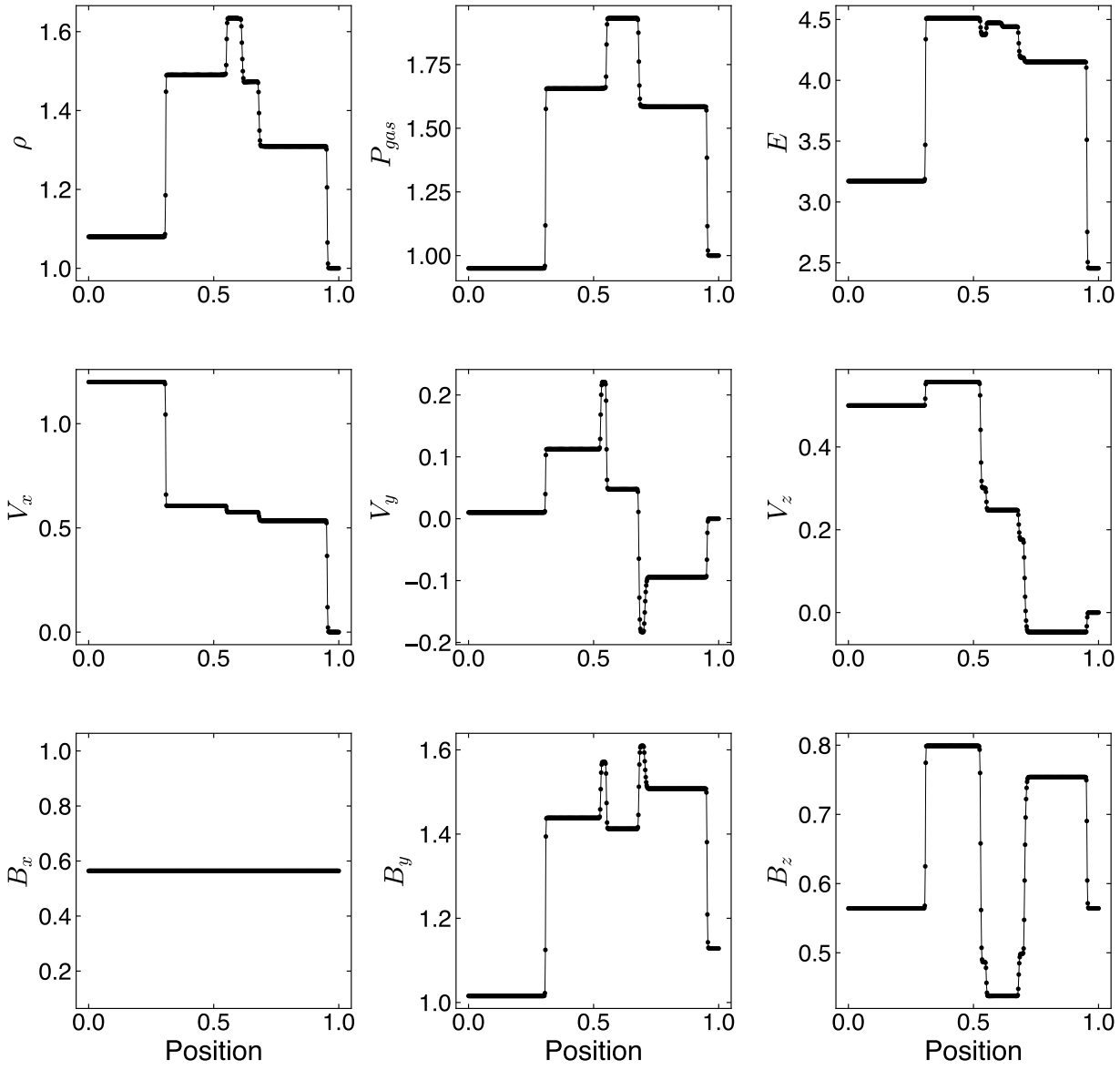

Figure 10: Dai & Woodward Shock Tube (also called Ryu & Jones 2a) solution [25, 95]. script link

4.1.4.3 Ryu & Jones 1a Shock Tube

Figure 11 shows the Ryu & Jones 1a Shock Tube solution [95] which is a less common test for MHD codes. However, in our experience, it is an excellent problem for debugging due to its relatively simple structure, which is easy to examine manually. The lack of any spikes and the presence of multiple types of strong shocks also make it a good diagnostic test for over/undershoot of the solution near discontinuities.

4.1.4.4 Ryu & Jones 4d Shock Tube

Figure 12 shows the Ryu & Jones 4d Shock Tube solution [95] which features a switch-on slow shock. Switch-on waves increase the strength of the transverse magnetic field while reducing the thermal pressure to maintain energy conservation. This is a simplified example of a type of magnetic field amplification and as such it is important to demonstrate that a code can replicate it accurately. A switch-off wave does the inverse.

4.1.4.5 MHD Einfeldt Strong Rarefaction

Figure 13 shows the MHD Einfeldt Strong Rarefaction test [30] which creates a strong outflow and central vacuum state. The diverging solution leads to an extremely strong and fast rarefaction where the energy is dominated by kinetic energy and as such can often reveal challenges for finite-volume methods with near-vacuum states, since some Riemann solvers will return unphysical solutions with negative density or negative internal energy. High values of the outflow velocity ($V_{out} \geq 3$) can also lead to spurious oscillations in the solution. $V_{out} = 2$ was chosen for this test [74]. Cholla performs well on this test with no spurious oscillations or unphysical negative values.

4.1.5 MHD Blast Wave in a Strongly Magnetized Medium

Blast waves in different forms are excellent tests for hydrodynamics and MHD codes. They combine strong shocked flows, smooth flows, and, in MHD, strong magnetic fields. The results are qualitative rather than quantitative, but thoroughly test the robustness of

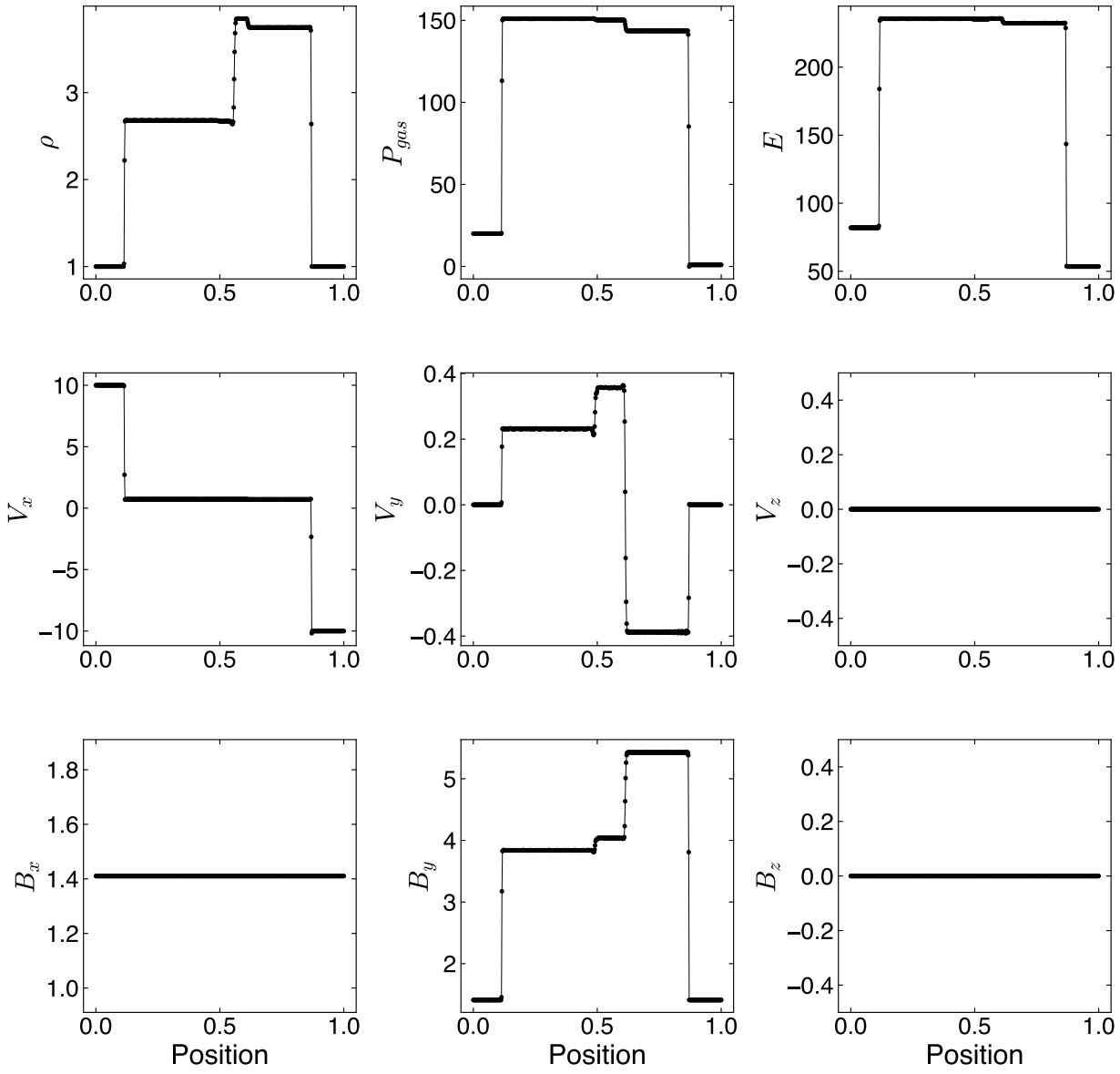

Figure 11: Ryu & Jones 1a Shock Tube solution [95]. script link

the algorithm and are excellent regression tests for automated testing (see Chapter 5). For this test we use $\beta = 0.2$; like [105], we find instabilities if β is decreased by a factor of 10.

The background state is $\rho = 1.0$, $P = 0.1$, $v_x = 0.0$, $v_y = 0.0$, $v_z = 0.0$, $B_x = 1/\sqrt{2}$, $B_y = 1/\sqrt{2}$, $B_z = 0.0$, and the over pressure region is a central sphere of size $R = 0.1$

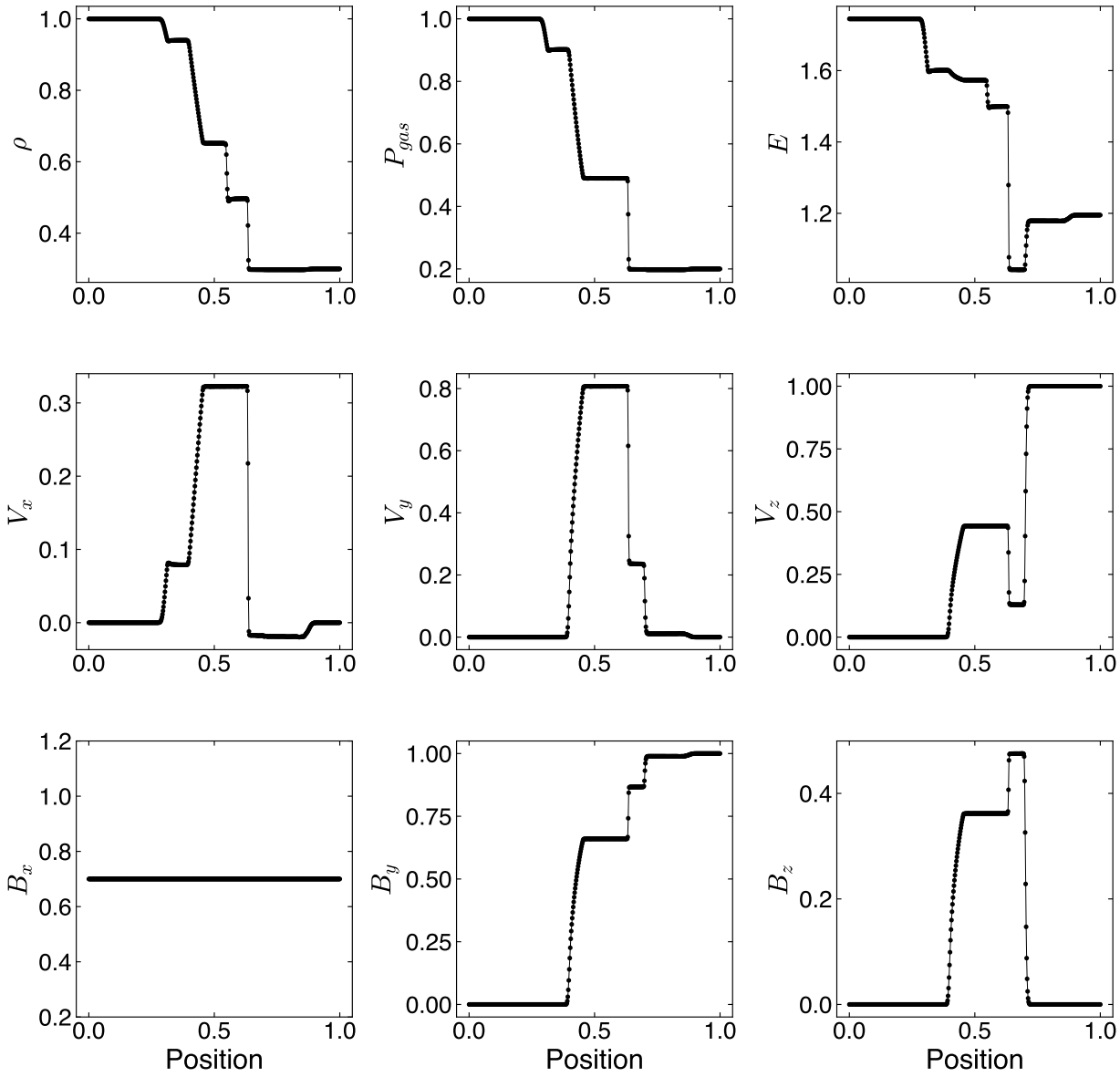

Figure 12: Ryu & Jones 4d Shock Tube solution [95]. script link

which has $P = 10.0$. The test is then run on a domain of $1 \times 1.5 \times 1$ with a resolution of $200 \times 300 \times 200$ cells until $t = 0.2$. Figure 14 shows contours of the density and magnetic energy fields in an $x-y$ slice through the center of the domain. The contours are smooth and symmetric and show clear elongation of the blast wave rarefaction parallel to the magnetic

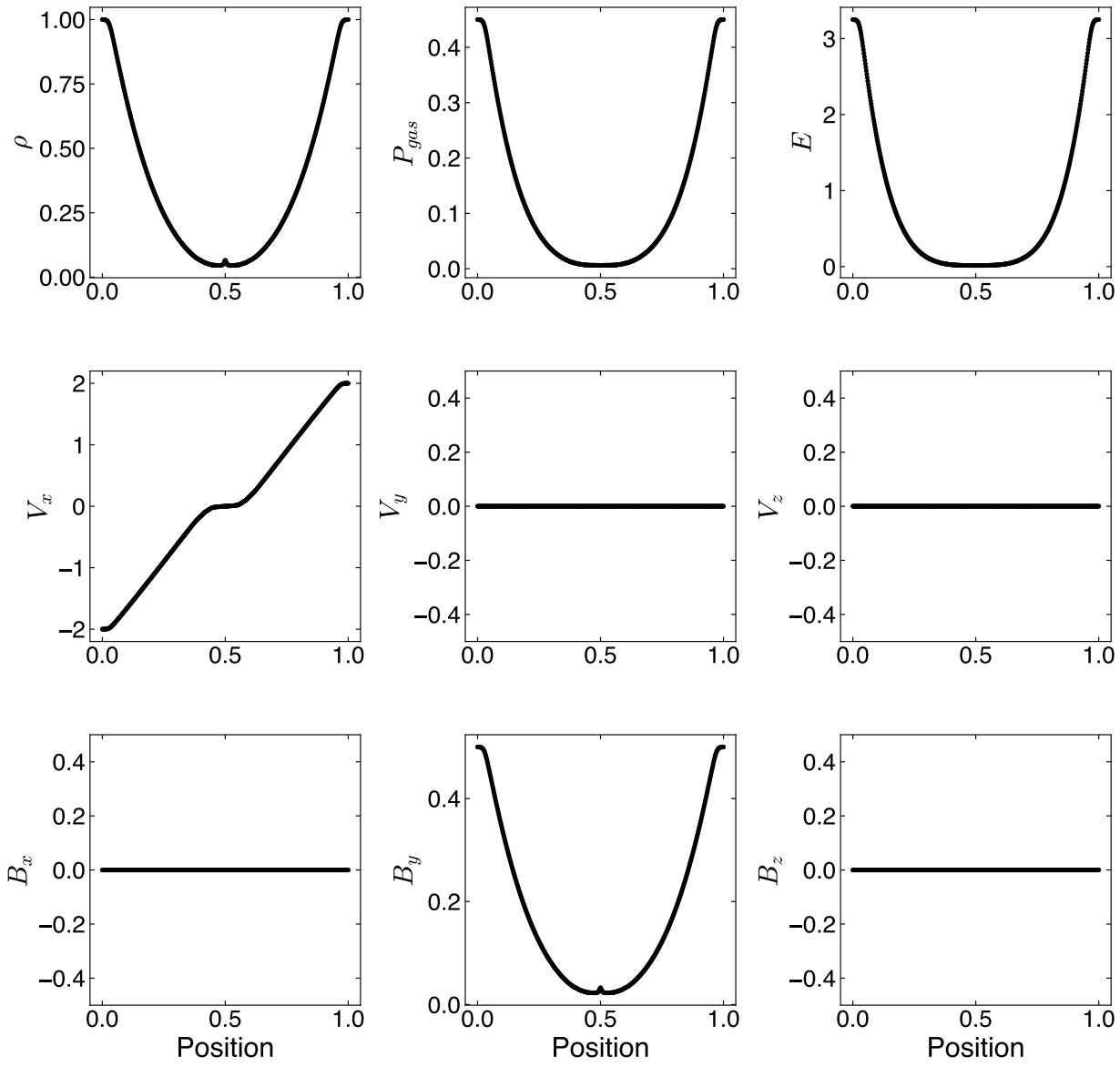

Figure 13: MHD Einfeldt Strong Rarefaction solution [30]. script link

field. The blast wave propagates slowly parallel to the magnetic field but much more rapidly perpendicular to the magnetic field.

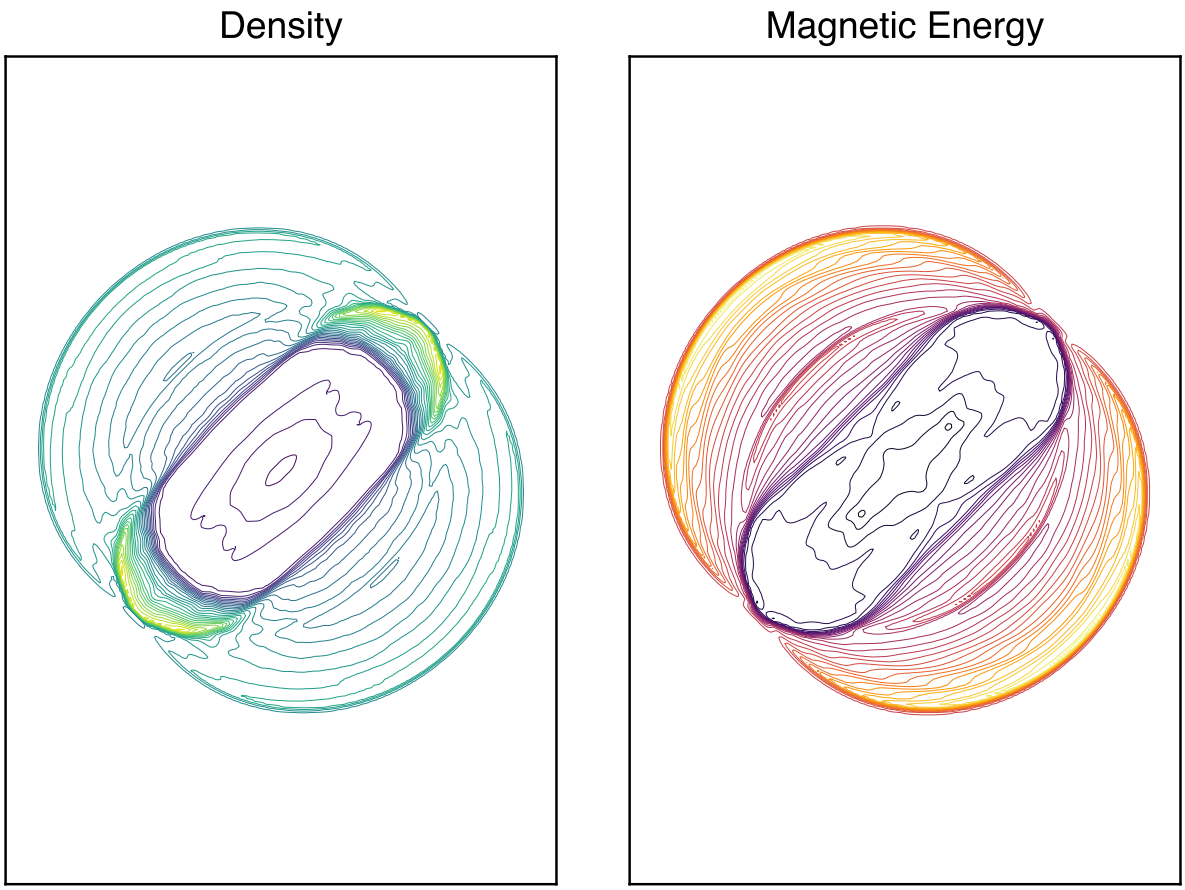

Figure 14: Contour plot of the MHD blast wave test at $t = 0.2$. 30 evenly spaced contours are shown in an $x - y$ slice through the center of the domain. script link

4.1.6 Orszag-Tang Vortex

The Orszag-Tang vortex is a standard 2D MHD test from [81]. While it does not provide a quantitative measure of accuracy like the linear wave tests or a test of the robustness of the method like the MHD blast wave, it does have a very complex flow that is sensitive to changes in the integrator, making it ideal for regression testing.

The test was conducted on a periodic domain of $1\times1\times1$ with a resolution of $192\times192\times192$ cells until $t = 0.5$ with the following initial conditions: $\rho = 25/(36\pi)$, $P = 5/(12\pi)$, $v_x = \sin 2\pi y$, $v_y = -\sin 2\pi x$, $v_z = 0.0$, $A_x = 0.0$, $A_y = 0.0$, $A_z = (B_0/4\pi)(\cos 4\pi x + 2\cos 2\pi y)$,

with $B_0 = 1\sqrt{4\pi}$. The results, plotted in Figure 15, can be compared directly to Figure 22 in [106] as a qualitative check for correctness of the flow structure.

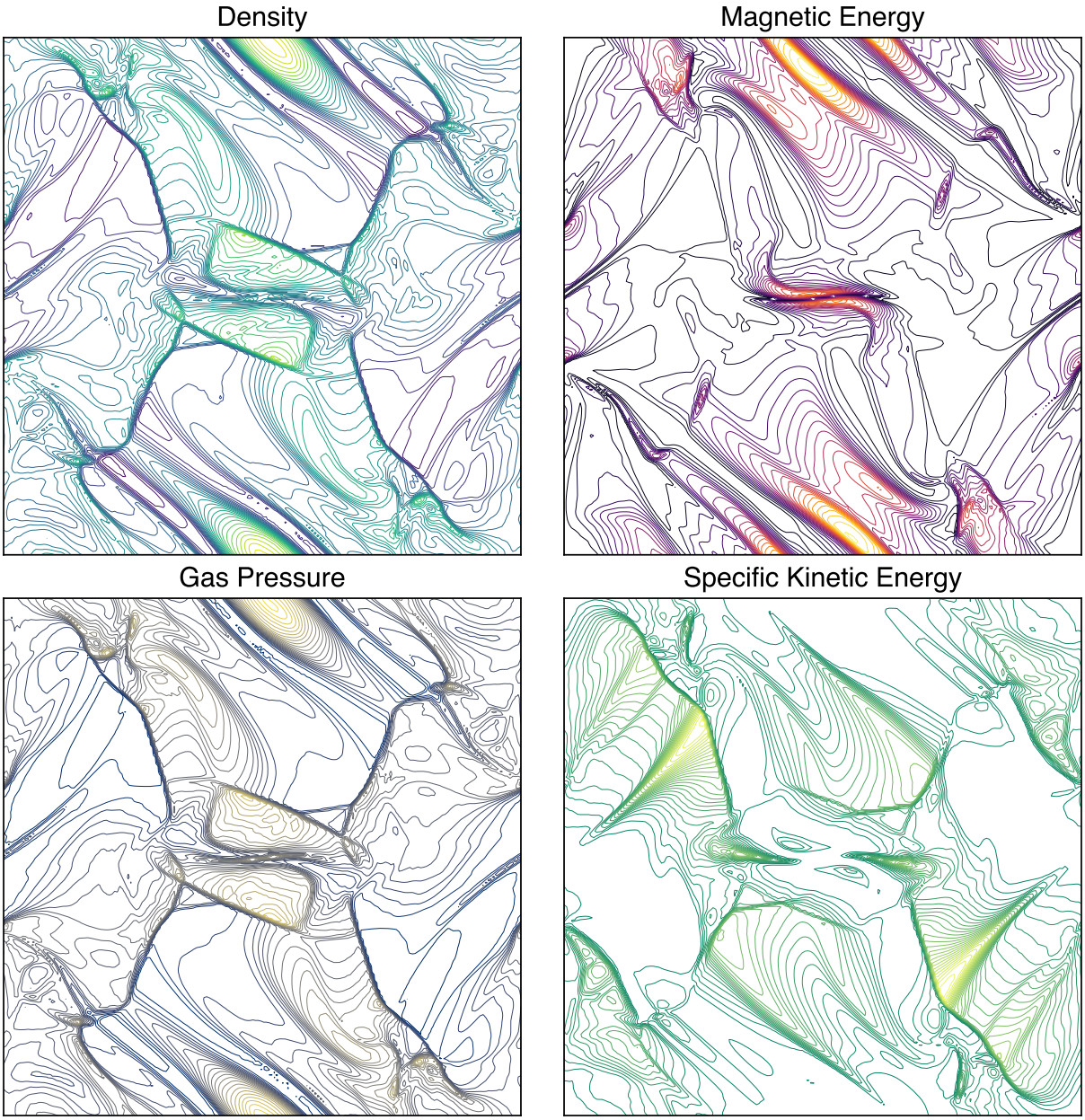

Figure 15: Contour plot of the Orszag-Tang Vortex at $t = 0.5$. Thirty evenly spaced contours are shown for each plot in an $x - y$ slice through the center of the domain. script link

4.2 MHD Performance Tests

Given that Cholla is a massively parallel code, its scaling properties warrant discussion. Our primary focus is on weak scaling rather than strong scaling, since good weak scaling enables much larger problems to be simulated, while strong scaling can lead to reducing the number of cells per GPU to the point where the whole GPU cannot be utilized, which will significantly impact performance. Results of our weak scaling tests are shown in Figure 16, while strong scaling is shown in Figure 17.

All of the weak scaling tests shown were performed with a slow magnetosonic wave perturbation (described in Section 4.1.1), periodic boundary conditions, in double precision, and with 459^3 cells per MPI rank; each rank is assigned one GPU. We employ the second order piecewise linear reconstruction method with limiting in the characteristic variables. The wave is evolved for 100 time steps (a wall-clock time of ~ 45 seconds) and the resulting time per step is averaged over the total number of time steps, excluding setup and tear down time. Strong scaling tests are performed with the same problem, though the wave is run through one full period, ~ 6100 time steps to ensure that it runs for a non negligible wall-clock time when large numbers of GPUs are utilized. The strong scaling test uses 459^3 cells, the most we can fit on a single MI250X GCD.

Our scaling tests were performed on the *Frontier* Supercomputer at the Oak Ridge Leadership Computing Facility. *Frontier* utilizes AMD MI250X GPUs, each of which contains two Graphics Compute Dies (GCDs) that largely function as separate GPUs and can be treated as such in software. Thus, for the sake of clear comparison to other systems, we will refer to each GCD as a single GPU for the remainder of this paper.

On *Frontier* Cholla updates 2.36×10^8 cells per second per GPU when running with a single GPU. The single-GPU performance is comparable on NVIDIA hardware: an NVIDIA V100 GPU achieved 160 million cell updates per second, and an A100 achieved 259 million cell updates per second. On 74,088 GPUs, nearly the entirety of *Frontier*, Cholla performs 1.89×10^8 cell updates per second per GPU, with a weak scaling efficiency of 82.2%. The 74,088 GPU run updated a total of 1.40×10^{13} cells per second on a total grid size of $19,278^3$ cells. This performance is comparable to other similar optimized GPU codes (see e.g. Figures

9 and 10 in [47]).[1]

Cholla's strong scaling performance (Figure 17) is close to ideal up to 32 GPUs (80% strong scaling efficiency) and does not drop below 50% until 256 GPUs. Strong scaling plateaus at around 512 GPUs, where the problem size per GPU is 58^3 cells. Running with a typical number of cells ($128^3 - 256^3$) to balance wall-time versus efficiency, we suffer at most a $\approx 55\%$ drop in efficiency. (In this strong scaling plot the closest problem size to 128^3 is the 64 GPU test, which has 115^3 cells per GPU and a strong scaling efficiency of 67.2%)

With a single MPI rank there is negligible communication overhead as no halo cells need to be exchanged between GPUs; the only boundary update required is to copy halo cell values from one location in GPU memory to another. This update is very fast, and accounts for $\sim 0.6\%$ of total runtime on a single rank. As the number of ranks grows the MPI overhead quickly stabilizes at around 15ms with a moderate increase when running on the full size of *Frontier*. Perhaps most importantly, the VL+CT integrator scales almost perfectly and takes up most of the time on each time step, dominating over the MPI communication time by nearly a factor of 10. The very slightly imperfect weak scaling of the integrator, which has no MPI communication, is due to GPU-to-GPU variance; as the number of GPUs increases the standard deviation in runtime for a single time step increases. Since all processes must wait for the slowest one to complete before beginning the next boundary exchange, this naturally leads to a few percent decrease in efficiency on these scales. Overall these tests demonstrate that Cholla has excellent weak scaling up to the full size of *Frontier*, over 74,000 GPUs.

[1]Note that the weak scaling plots in [47] are normalized to single node performance, not single GPU performance, and they report performance for a 2nd order hydrodynamic solver, not MHD.

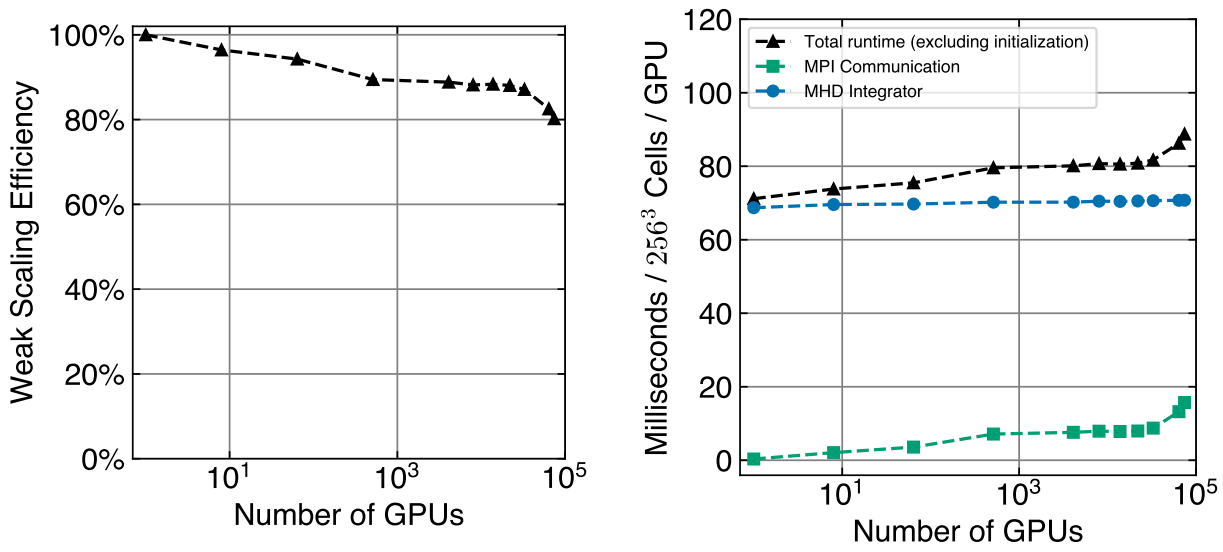

Figure 16: Weak scaling performance of Cholla MHD. When running on a single GPU Cholla updates 2.04×10^8 cells per second per GPU; the largest run with 74,088 GPUs updates 1.67×10^8 cells per second per GPU, a weak scaling efficiency of 82.2%. The 74,088 GPU run updates a total of 1.24×10^{13} cells per second. script link

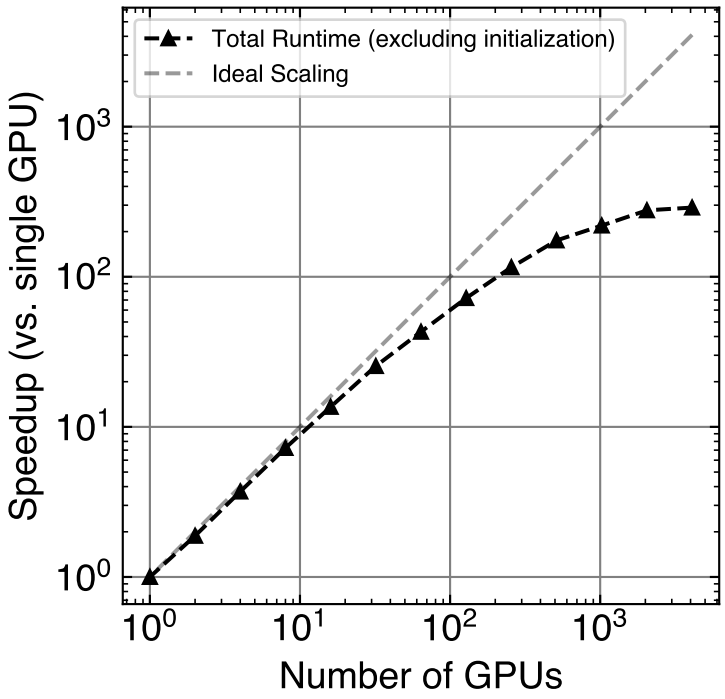

Figure 17: Strong scaling performance of Cholla MHD with a problem size of 459^3 cells. script link

5.0 Automated Testing & Continuous Integration

As Cholla has continued to grow in complexity, and with the continued addition of large new physics modules like MHD that require changes to much of the code base, the need for a more robust, automated testing system has become increasingly apparent. Several challenges exist in implementing such a system for Cholla - not only must the testing system accommodate GPU hardware, but it must also work on large parallel systems like *Frontier*. We have addressed this need in three primary steps: 1) Choosing a testing framework for unit tests, 2) Writing the software tools needed to extend that testing framework for Cholla's needs, and 3) adding automated testing as part of a continuous integration (CI) pipeline that runs whenever anyone submits a pull request to the Cholla GitHub repository. This Chapter describes the implementation of our automated testing and continuous integration framework.

Taken together, the ability to add tests for any part of the code and their automated running on every pull request has meant that new features are faster and easier to add to Cholla with confidence. Errors or changes that may otherwise break older code are often caught by the tests before they ever make it into the code base.

5.1 Unit Testing Framework

Three main kinds of tests are needed for scientific code bases: unit tests, integration tests, and system tests (also called "end-to-end" tests). Unit tests check a single "unit" of code, i.e. a single function, class method, data structure, etc. Integration tests check how units of code operate together; for example, a test of the HLLD solver is an integration test because the solver internally calls many different functions and uses multiple different data structures. System tests check the entire code base, often with a simple test problem like a Riemann problem or linear wave, and verify that the entire program produces the correct output.

We chose GoogleTest[1] for the unit testing framework due to its large number of features, general popularity, and relative ease of use. Another primary requirement was a testing framework that supports "death tests", i.e. tests that check if the internals of the test crash/segfault/etc. Since Cholla, like many high performance computing (HPC) codes, handles errors by reporting those errors and then exiting, a testing framework that can handle code crashes without crashing the tests as well is critical. GoogleTest is available already built on many HPC systems and, if it is not available, can be built as an optional part of the test running script.

5.2 Extensions for Cholla

Two primary extensions were required for fully testing Cholla: a robust method for comparing floating point numbers and a way to run system tests.

5.2.1 Floating Point Comparisons

In order to run either unit tests or system tests, the code must have a robust method for comparing floating point numbers for equality. Comparing floating point numbers for equality is notoriously challenging and generally the exact comparison method that should be used varies by application[45, 76][2]. Absolute comparisons ($|a - b| < X$) work well for small numbers, but with larger numbers the difference between two successive floats can be much larger than a typical value for X and therefore a different comparison method is required. We chose a hybrid method of both an absolute comparison and a Units in Last Place (ULP) comparison. A ULP comparison determines how many representable floating point numbers there are between any two floats. By default the hybrid method we employ first performs an absolute check, $|a - b| < 10^{-14}$. This number was chosen as we found that typical differences in the Sod Shock Tube solution when comparing results with different hardware and compilers resulted in differences of $\sim 5 \times 10^{-15}$. After the absolute check a

[1] https://github.com/google/googletest

[2] https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/

ULP check is performed with a maximum allowed error of 4. If either check passes then the numbers are deemed to be "equal". Having a robust method to compare floating point numbers is critical since the output of a code is not guaranteed to be bitwise identical when the code is compiled with different compilers, run on different systems, etc.

5.2.2 System Tests

GoogleTest provides most of the tools required to run unit and integration tests, but validating the results of an entire system test requires additional infrastructure. Running a system test requires launching the program to test with correct initial conditions, checking that the program did not crash, loading both the generated data to test and the fiducial data, then comparing those two data sets. In order to perform system tests with Cholla, we added a class that performs all of the required tasks, which include launching Cholla with any number of MPI ranks as well as comparing the results against fiducial data. To facilitate running across a wide range of MPI ranks and on clusters with queue systems, the class is designed to allow system tests to be run in different modes: one can either launch Cholla and save the results, compare already existing test data to fiducial data, or do both. This enables the user to run Cholla on many thousands of ranks then later launch a separate job to make the actual comparison. We have found that on up to 10,000 ranks, with small simulation grids (typically $< 64^3$ cells) per rank the latter comparison only takes a few minutes on a single CPU core. Most of the time, however, both steps can be run within the same job, since large tests with many ranks are not required for most development work.

Two primary methods of comparison are used to determine the success of a system test. These are either a direct cell-by-cell comparison of the results for each field using the floating point comparison tools described above (Section 5.2.1), or a calculation of the L2 norm of the L1 error vector as described in Section 4.1.1. The cell-by-cell comparison is quite accurate, but can be fragile on some complex tests if a small number of cells have errors that are slightly larger than typical, which can lead to false failures when comparing results between systems or compilers. The L2 norm method is less fragile to small errors in a handful of cells, but is generally less sensitive, so we only use it on the tests where it is required, namely the

MHD blast wave (subsection 4.1.5) and advecting field loop (subsection 4.1.3).

5.3 Automated Testing

To ensure that these tests are run regularly and all new code is tested we have made the existing tests as easy to run as possible, and we require that they are all run automatically on each pull request. To facilitate this, Cholla's build directory includes a script which performs all required setup, installs GoogleTest (if requested), builds Cholla, and then runs all the tests. The script also includes a function that combines all of these into a single function call for ease of use, for example, if a user is running tests manually (say, prior to submitting a PR).

Implementing automated testing for HPC codes is not always an easy task. Automated tests are an aspect of Continuous Integration (CI) – the practice of automating the addition of code changes and additions from a team of developers into a software project. Cholla is currently designed to run on CUDA or HIP capable GPUs, so GPU hardware is required in order to incorporate continuous integration (CI), something that few current CI services offer at a reasonable cost for academic users. Many of Cholla's physics modules are turned on and off at compile time, which presents another challenge, since not all code will be tested for every compilation configurations. Similarly, we need to be able to ensure that Cholla works properly on both AMD and NVIDIA GPUs, which require different compilation targets. This means that there is no single binary file that contains all of Cholla. Instead we have multiple "builds" that each require testing. While this does result in a high performance executable that only contains the necessary code, it makes testing the code much more complex due to the number of possible build configurations.

Our solution to these issues was to use a mix of two different systems. When a pull request is submitted to the Cholla GitHub repository several jobs are launched: A GitHub Actions[3] job to check code formatting, a GitHub Actions matrix job to build all the HIP/AMD builds

[3] `https://github.com/features/actions`

using a Docker container, and a Jenkins[4] matrix job running on local hardware that runs the CUDA/NVIDIA builds, tests, and static analyzers. Thus, every common configuration of Cholla is built with both CUDA and HIP on every pull request and the CUDA builds are also tested to ensure that no existing or new tests fail.

[4] `https://www.jenkins.io`

6.0 Summary

We have presented the MHD extension to Cholla, a massively parallel, GPU native, astrophysical simulation code. MHD in Cholla uses the Van Leer plus Constrained Transport MHD integrator (VL+CT) [105], the HLLD Riemann solver [75], and includes multiple high order reconstruction methods to model numerical solutions to the Eulerian ideal MHD equations on a static mesh.

We showed the modifications required to implement MHD on GPUs compared to CPUs and discussed challenges working within the limits of GPUs in Section 3.3. One major challenge was moving computational work and data storage from the CPU to the GPU. Previous versions of Cholla did some of the computation on the CPU and used CPU memory to effectively expand GPU memory. This required regularly copying data between the CPU and the GPU which became a performance bottleneck. The current version of Cholla maintains all the data, and the vast majority of the work, on the GPU which has led to a considerable speedup. As demonstrated in Section 4.2, these optimizations combined with the highly parallel nature of GPUs make MHD in Cholla extremely fast, with 259 million cell updates per GPU-second when running on a single NVIDIA A100. Cholla also demonstrates excellent weak scaling, and achieves a weak scaling efficiency of 80% when scaled up to 74,088 GPUs on *Frontier* which utilizes AMD MI250X GPUs; with a total of 14.0 trillion cell updates per second (see Figure 16).

We have also presented a suite of canonical MHD tests in Chapter 4. These tests demonstrate the accuracy of Cholla across a broad range of problems. They also demonstrate that the VL+CT MHD algorithm does an excellent job of maintaining the divergence free condition even in highly challenging settings.

To accommodate the increasing complexity of Cholla and facilitate multiple simultaneous development efforts, we have also added a structured testing framework, described in Chapter 5. This framework is based on GoogleTest, augmented with custom testing tools. This approach enables running tests that range from single function unit tests to massively parallel system tests across the scale of an entire cluster within the same framework. We have further

integrated these tests with GitHub Actions to run formatting, static analysis, and builds with various physics configurations along with Jenkins running on local resources.

Cholla is free and open source software available at `https://github.com/cholla-hydro/cholla`. In general, Cholla is designed to be flexible and modular, and can be run with or without the new MHD module. In addition, the Cholla framework can easily accommodate additional physics modules in the future, some of which are in progress. We welcome new development efforts, and hope that this work will be a resource for the broader astrophysics simulation community.

6.1 Application of Cholla MHD

One potential application of Cholla-MHD is to perform a series of global galaxy simulations similar to the CGOLS suite[99] but with the inclusion of MHD. The goal would be to study how magnetic draping of cold clouds in the CGM impacts their survival, size, velocity, and overall dynamics of the CGM. Magnetic draping is when an object, in this case a cold cloud, moves super-Alfénically through a magnetic field and sweeps up a substantial magnetic layer which is "draped" over the cloud. This strong magnetic layer can inhibit thermal conduction and mixing, potentially allowing a draped cloud to survive much longer in a hot wind. While this phenomenon has been studied in idealized cloud-wind simulations [2, 49, 14, 57] and, very recently, in cosmological simulations [93], it has never been studied in high resolution in galactic outflow simulations. The results of cloud-wind simulations depend strongly on the initial orientation of the magnetic field. In Cholla-MHD simulations the magnetic fields in the clouds would not be set as part of the initial conditions but would instead arise organically through the dynamics of the disk and CGM, leading to much more realistic launching conditions of the outflows. While the cosmological simulations have similar in situ magnetic field orientations, their physical resolution is much lower due to the larger regions being simulated. We would be able to run a 1 trillion cell simulation while utilizing approximately half to 2/3rds of *Frontier*, which would allow us to simulate cloud launching and the impact of magnetic draping with realistic in situ magnetic fields and very

high resolution in the CGM. The initial conditions we would follow the CGOLS initial conditions with the addition of a magnetic field. The CGOLS simulations initial conditions are:

- Domain: $(L_x, L_y, L_z) = (10kpc, 10kpc, 20kpc)$
- Disk: A disk of isothermal gas at 10^4K with an exponential surface density of profile and a central surface density of $150 M_{\odot}\text{pc}^{-2}$
- Halo Gas: A hot static halo in hydrostatic equilibrium
- Gravity: A constant gravitational potential for the dark matter halo using the Navarro-Frenk-White profile
- Magnetic Fields: Varying depending on the simulation in the suite. See below.

In addition to MHD we would use several of the other physics modules in Cholla to accurately simulate a galaxy; namely:

- Feedback: A stellar feedback model based on Kim & Ostriker 2015 [64] which injects energy or momentum into the ISM.
- Gravity: In addition to the static dark matter gravitational potential, an FFT based self gravity for the baryonic gas. This is a major departure from the CGOLS simulations which utilize a static potential for both the dark matter and the baryonic gas.
- Cooling: A simple radiative cooling model based on Cloudy [35]
- Dual Energy: Given the extremely high velocities of gas in the outflow, tracking the internal energy of the gas separately from its kinetic energy significantly improves accuracy.

We could then run a suite of simulations with various magnetic field strengths and morphologies to examine the impact on the morphology and dynamics of the CGM: A "baseline" or "control" simulation run with MHD turned on but with the magnetic field set to zero to compare the versions with magnetic fields to and two simulations with magnetic fields. Both simulations with magnetic fields would have initial magnetic field with a strength similar to what is observed in starburst galaxies like M82, $\approx 50 - 100\mu G$. The first MHD simulation would have a a simple initial magnetic field with the field perpendicular to the

disk. We do not expect the initial strength or morphology of the initial magnetic field to have a significant impact on the morphology or relative growth rate of the magnetic field after the first $\approx$ 25Myrs of evolution [70, 37]. However, we are interested in studying the dynamics of an evolved galaxy, not the evolution of the galaxy, and those dynamics can be significantly impacted by the dynamics in the first $\approx$ 25Myrs of a simulation. As such we would want to run a second simulation with a turbulent initial magnetic field and contrast it with the simple magnetic field to see if the initial conditions of the magnetic field have a significant impact. These simulations would allow us to examine the role of magnetic fields in galactic outflows and specifically what the impact, if any, magnetic draping plays in cloud survival, growth, acceleration, and morphology.

Appendix HLLD MHD Riemann Solver

This is a brief summary of the HLLD Riemann solver I implemented into Cholla. The HLLD Riemann solver was originally introduced in [75] and since then has become the standard Riemann solver for MHD. It is very similar in structure to the existing HLLC Riemann solver with the inclusion of magnetic fields and the additional MHD waves that they bring. Note that this is an approximate Riemann solver and one of the primary approximations is that it neglects the slow magnetosonic wave. Nonetheless, the tests is Section 4 show that it can accurately approximate the slow magnetosonic wave (see Section 4.1.1) and even slow switch-on shocks (see the Ryu & Jones 4D shock tube in Section 4.1.4).

The HLLD Riemann solver operates in one of six states, the left/right states, the left/right star states, and the left/right double star states. Throughout this discussion the state of a specific variable will be indicate with an subscript to indicate left/right, no superscript to indicate the non-star state, a single asterisk superscript to indicate the star state, and a double asterisk superscript to indicate the double star state, all other variable definitions are consistent with earlier definitions.

A.1 Compute Acoustic & Contact Wave Speeds

The first step is to compute the $S_L, S_L^*, S_M, S_R^*, \text{and} S_R$ wave speeds. These wave speeds are approximations to the speeds of the left and right moving fast magnetosonic waves (S_L and S_R), the left and right moving Alfvén wave speeds (S_L^* and S_R^*), and the speed of the contact wave (S_M). These wave speeds are used to choose the state of the Riemann solver which in turn determines the correct fluxes.

A.1.1 Computing S_L and S_R

We can use either of the below approximations for S_L and S_R (there are other options as well, these were just the two given in [75]).

$$S_L = \min(\lambda_{min}(\vec{U}_L), \lambda_{min}(\vec{U}_R)) \tag{59}$$

$$S_R = \max(\lambda_{max}(\vec{U}_L), \lambda_{max}(\vec{U}_R)) \tag{60}$$

Where λ_{min} is the smallest eigenvalue and λ_{max} is the largest.
or

$$S_L = \min(v_{x,L}, v_{x,R}) - \max(c_{f,L}, c_{f,R}) \tag{61}$$

$$S_R = \max(v_{x,L}, v_{x,R}) + \max(c_{f,L}, c_{f,R}) \tag{62}$$

Where $c_{f,L}$ and $c_{f,R}$ are the fast magnetosonic wave speeds in the left and right cells respectively. In Cholla we use Equations 61 and 62 to compute the approximate wave speeds as we have founds them slightly more stable and simpler to implement.

A.1.2 Computing S_L^* and S_R^*

Note that the second term is the Alfvén speed in the star state.

$$S_L^* = S_M - \frac{\mid B_x \mid}{\sqrt{\rho_L^*}} \tag{63}$$

$$S_R^* = S_M + \frac{\mid B_x \mid}{\sqrt{\rho_R^*}} \tag{64}$$

where

$$\rho_k^* = \rho_k \frac{S_k - v_{x,k}}{S_k - S_M} \tag{65}$$

A.1.3 Computing S_M

$$S_M = \frac{\rho_R v_{x,R}\left(S_R - v_{x,R}\right) - \rho_L v_{x,L}\left(S_L - v_{x,L}\right) + p_{T_L} - p_{T_R}}{\rho_R\left(S_R - v_{x,R}\right) - \rho_L\left(S_L - v_{x,L}\right)} \tag{66}$$

A.2 Determine the State

Use the equation below with the wave speeds to determine the state of the Riemann solver and the next three sections handle the non-star, star, and double star states respectively.

$$\vec{F}_{HLLD} = \begin{cases} \vec{F}_L & \text{if } 0 < S_L \\ \vec{F}_L^* & \text{if } S_L \leq 0 < S_L^* \\ \vec{F}_L^{**} & \text{if } S_L^* \leq 0 < S_M \\ \vec{F}_R^{**} & \text{if } S_M \leq 0 < S_R^* \\ \vec{F}_R^* & \text{if } S_R^* \leq 0 \leq S_R \\ \vec{F}_R & \text{if } S_R < 0 \end{cases} \tag{67}$$

A.3 Compute & Return the Fluxes

A.3.1 F_L or F_R State

This is the region outside of the fast magnetosonic wave. The fluxes are:

$$\vec{F}_k = \begin{bmatrix} \rho_k v_{x,k} \\ \rho_k v_{x,k}^2 + p_{T,k} - B_{x,k}^2 \\ \rho v_{x,k} v_{y,k} - B_{x,k} B_{y,k} \\ \rho v_{x,k} v_{z,k} - B_{x,k} B_{z,k} \\ 0 \\ B_{y,k} v_{x,k} - B_{x,k} v_{y,k} \\ B_{z,k} v_{x,k} - B_{x,k} v_{z,k} \\ v_{x,k} \left(E_k + p_{T,k}\right) - B_{x,k} \left(\vec{v_k} \cdot \vec{B_k}\right). \end{bmatrix} \tag{68}$$

A.3.2 F_L^* or F_R^* State

This is the region between the fast magnetosonic waves and the Alfvén waves. These fluxes are more complicated and require the use of the fluxes from the non-star state as well. The algorithm is exactly the same for the left and right star states so the subscript k is used to indicate R or L.

$$\vec{F}_k^* = \vec{F}_k + S_k \left(\vec{U}_k^* - \vec{U}_k\right) \tag{69}$$

and we find the components of $\vec{U}_k^*$ with the following equations. ρ_k^* was already computed when finding S_k^* and $v_{x,k}^* = S_M$.

Pressure is a bit more complicated. [75] show that, under their assumptions, $p_{T,L}^* = p_{T,L}^{**} = p_{T,R}^* = p_{T,R}^{**} = p_T^* = p_T^{**}$

$$p_T^* = p_{T,L} + \rho_L \left(S_L - v_{x,L}\right)\left(S_M - v_{x,L}\right) = p_{T,R} + \rho_R \left(S_R - v_{x,R}\right)\left(S_M - v_{x,R}\right) \tag{70}$$

They also note that

$$p_T^* = \frac{\rho_R p_{T_L} \left(S_R - v_{x,R}\right) - \rho_L p_{T_R} \left(S_L - v_{x,L}\right) + \rho_L \rho_R \left(S_R - v_{x,R}\right)\left(S_L - v_{x,L}\right)\left(v_{x,R} - v_{x,L}\right)}{\rho_R \left(S_R - v_{x,R}\right) - \rho_L \left(S_L - v_{x,L}\right)}. \tag{71}$$

And so

$$E_k^* = \frac{E_k\left(S_k - v_{x,k}\right) - p_{T,k}v_{x,k} + p_T^* S_M + B_x\left(\vec{v}_k \cdot \vec{B}_k - \vec{v}_k^* \cdot \vec{B}_k^*\right)}{S_k - S_M}. \tag{72}$$

We can find the y and z components of $\vec{v}$ and $\vec{B}$ with the following equations. Note these equations can give a term that is $\frac{0}{0}$ if $S_M = v_{x,k}$, $S_k = v_{x,k} \pm c_{f,k}$, $B_{y,k} = B_{z,k} = 0$, and $B_x^2 \geq \gamma p_k$. If this is the case then there is no shock across S_k, so set $\vec{v}_k^* = \vec{v}_k$, $B_{y,k} = B_{z,k} = 0$, $\rho_k^* = \rho_k$, and $p_{T,k}^* = p_{T,k}$.

$$v_{j,k}^* = v_{j,k} - B_x B_{j,k}\frac{S_M - v_{x,k}}{\rho_k\left(S_k - v_{x,k}\right)\left(S_k - S_M\right) - B_x^2} \tag{73}$$

$$B_{j,k}^* = B_{j,k}\frac{\rho_k\left(S_k - v_{x,k}\right)^2 - B_x^2}{\rho_k\left(S_k - v_{x,k}\right)\left(S_k - S_M\right) - B_x^2} \tag{74}$$

Note that the denominators are the same.

A.3.3 F_L^{**} or F_R^{**} State

This is the region between the Alfvén waves and the contact discontinuity (entropy wave). These fluxes require the fluxes from both the previous steps. The algorithm is exactly the same for the left and right star states so the subscript k is once again used to indicate R or L.

$$\vec{F}_k^{**} = \vec{F}_k^* + S_k^*\left(\vec{U}_k^{**} - \vec{U}_k^*\right) \tag{75}$$

We already know some of the state variables from previous computations

$$v_{x,k}^{**} = S_M \tag{76}$$

$$p_T^{**} = p_T^* \tag{77}$$

$$\rho_k^{**} = \rho_k^* \tag{78}$$

So all we need to compute directly is v_y^{**}, v_z^{**}, B_y^{**}, B_z^{**}, and E_k^{**}

$$v_j^{**} = \frac{v_{j,L}^* \sqrt{\rho_L^*} + v_{j,R}^* \sqrt{\rho_R^*} + \left(B_{j,R}^* - B_{j,L}^*\right) \mathrm{sign}\left(B_x\right)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \tag{79}$$

$$B_j^{**} = \frac{B_{j,R}^* \sqrt{\rho_L^*} + B_{j,L}^* \sqrt{\rho_R^*} + \left(v_{j,R}^* - v_{j,L}^*\right) \sqrt{\rho_L^* \rho_R^*} \mathrm{sign}\left(B_x\right)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \tag{80}$$

Note that the denominators are the same as is one of the coefficients in each term in the numerator.

Lastly we compute the energy (the minus and plus correspond to the L and R side respectively)

$$E_k^{**} = E_k^* \mp \sqrt{\rho_k^*} \left(\vec{v}_k^* \cdot \vec{B}_k^* - \vec{v}_k^{**} \cdot \vec{B}_k^{**}\right) \mathrm{sign}\left(B_x\right) \tag{81}$$

Bibliography

[1] Ann Almgren, Maria Sazo, John Bell, Alice Harpole, Max Katz, Jean Sexton, Donald Willcox, Weiqun Zhang, and Michael Zingale. CASTRO: A Massively Parallel Compressible Astrophysics Simulation Code. *Journal of Open Source Software*, 5(54):2513, October 2020.

[2] W. E. Banda-Barragán, E. R. Parkin, C. Federrath, R. M. Crocker, and G. V. Bicknell. Filament formation in wind-cloud interactions - I. Spherical clouds in uniform magnetic fields. *Monthly Notices of the Royal Astronomical Society*, 455(2):1309–1333, January 2016.

[3] W. E. Banda-Barragán, E. R. Parkin, C. Federrath, R. M. Crocker, and G. V. Bicknel. Filament formation in wind-cloud interactions - I. Spherical clouds in uniform magnetic fields. *Monthly Notices of the Royal Astronomical Society*, 455(2):1309–1333, 2016. Publisher: Oxford University Press.

[4] Wladimir Banda-Barragán, Christoph Federrath, Roland Crocker, and Geoffrey Bicknell. Filament formation in wind-cloud interactions. II. Clouds with turbulent density, velocity, and magnetic fields. *MNRAS*, 473:3454–3489, June 2017.

[5] Paul Batten, Nicholas Clarke, Claire Lambert, and Derek M Causon. On the choice of wavespeeds for the hllc riemann solver. *SIAM Journal on Scientific Computing*, 18(6):1553–1570, 1997.

[6] K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[7] Rainer Beck. Magnetic fields in spiral galaxies. *The Astronomy and Astrophysics Review*, 24(1):4, December 2016.

[8] Rainer Beck, Axel Brandenburg, David Moss, Anvar Shukurov, and Dmitry Sokoloff. GALACTIC MAGNETISM: Recent Developments and Perspectives. *Annual Review of Astronomy and Astrophysics*, 34(1):155–206, September 1996.

[9] Dave Binkley, Marcia Davis, Dawn Lawrie, and Christopher Morrell. To camelcase or under_score. In *2009 IEEE 17th International Conference on Program Comprehension*, pages 158–167, 2009.

[10] Roger Blandford, David Meier, and Anthony Readhead. Relativistic jets from active galactic nuclei. *Annual Review of Astronomy and Astrophysics*, 57(1):467–509, 2019.

[11] Jeremiah U Brackbill and Daniel C Barnes. The effect of nonzero $\nabla \cdot b$ on the numerical solution of the magnetohydrodynamic equations. *Journal of Computational Physics*, 35(3):426–430, 1980.

[12] Axel Brandenburg and Evangelia Ntormousi. Galactic dynamos. *Annual Review of Astronomy and Astrophysics*, 61(1):561–606, 2023.

[13] M Brio and C C Wu. An Upwind Differencing Scheme for the Equations of Ideal Magnetohydrodynamics. *JOURNAL OF COMPUTATIONAL PHYSICS*, 75:40–422, 1988.

[14] Marcus Brüggen, Evan Scannapieco, and Philipp Grete. The Launching of Cold Clouds by Galaxy Outflows. V. The Role of Anisotropic Thermal Conduction. *The Astrophysical Journal*, 951(2):113, July 2023.

[15] Marcus Brüggen, Evan Scannapieco, and Philipp Grete. The Launching of Cold Clouds by Galaxy Outflows. V. The Role of Anisotropic Thermal Conduction. *The Astrophysical Journal*, 951(2):113, July 2023.

[16] Tobias Buck, Christoph Pfrommer, Rüdiger Pakmor, Robert J J Grand, and Volker Springel. The effects of cosmic rays on the formation of Milky Way-mass galaxies in a cosmological context. *Monthly Notices of the Royal Astronomical Society*, 497(2):1712–1737, September 2020.

[17] D. Bégué, A. Pe'er, G.-Q. Zhang, B.-B. Zhang, and B. Pevzner. cuHARM: A New GPU-accelerated GRMHD Code and Its Application to ADAF Disks. *The Astrophysical Journal Supplement Series*, 264(2):32, February 2023.

[18] Yann Carteret, Abhijit B. Bendre, and Jennifer Schober. Observational Signatures of Galactic Turbulent Dynamos, August 2022. arXiv:2208.14178 [astro-ph].

[19] T K Chan, D Kereš, P F Hopkins, E Quataert, K-Y Su, C C Hayward, and C-A Faucher-Giguère. Cosmic ray feedback in the FIRE simulations: constraining cosmic ray propagation with GeV γ-ray emission. *Monthly Notices of the Royal Astronomical Society*, 488(3):3716–3744, September 2019.

[20] Paul Charbonneau. Solar dynamo theory. *Annual Review of Astronomy and Astrophysics*, 52(1):251–290, 2014.

[21] Phillip Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87(1):171–200, March 1990.

[22] Phillip Colella and Paul R Woodward. The Piecewise Parabolic Method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201, April 1984.

[23] R Courant, K Friedrichs, and H Lewyt. On the Partial Difference Equations of Mathematical Physics. *IBM Journal*, March 1967.

[24] Robert K. Crockett, Phillip Colella, Robert T. Fisher, Richard I. Klein, and Christopher F. McKee. An unsplit, cell-centered godunov method for ideal mhd. *Journal of Computational Physics*, 203(2):422–448, 2005.

[25] Wenlong Dai and Paul R Woodward. On The Diverrgence-Free Condition and Conservation Laws in Numerical Simulations for Supersonic Magnetohydrodynamic Flows. *THE ASTROPHYSICAL JOURNAL*, 494:317–335, 1998.

[26] Shane W. Davis and Alexander Tchekhovskoy. Magnetohydrodynamics simulations of active galactic nucleus disks and jets. *Annual Review of Astronomy and Astrophysics*, 58(1):407–439, 2020.

[27] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic Divergence Cleaning for the MHD Equations. *Journal of Computational Physics*, 175(2):645–673, January 2002.

[28] L. Del Zanna, M. Velli, and P. Londrillo. Parametric decay of circularly polarized Alfvén waves: Multidimensional simulations in periodic and open domains. *Astronomy & Astrophysics*, 367(2):705–718, February 2001.

[29] C. L. Dobbs and D. J. Price. Magnetic fields and the dynamics of spiral galaxies: Magnetic fields and spiral galaxies. *Monthly Notices of the Royal Astronomical Society*, 383(2):497–512, December 2007.

[30] B Einfeldt, C D Munz, P L Roe, and B Wgreen. On Godunov-Type Methods near Low Densities. *JOURNAL OF COMPUTATIONAL PHYSICS*, 92:213–295, 1991.

[31] Charles R Evans and John F Hawley. Simulation of Magnetohydrodynamic Flows: A Constrained Transport Model. *Astrophysical Journal*, 332:659–677, 1988.

[32] S. A. E. G. Falle. Self-similar jets. *Monthly Notices of the Royal Astronomical Society*, 250(3):581–596, June 1991.

[33] Marion Farcy, Joakim Rosdahl, Yohan Dubois, Jérémy Blaizot, and Sergio Martin-Alvarez. Radiation-magnetohydrodynamics simulations of cosmic ray feedback in disc galaxies. *Monthly Notices of the Royal Astronomical Society*, 513(4):5000–5019, 04 2022.

[34] Kyle Gerard Felker and James Stone. A fourth-order accurate finite volume method for ideal MHD via upwind constrained transport. *Journal of Computational Physics*, 375:1365–1400, December 2018. arXiv:1711.07439 [astro-ph, physics:physics].

[35] Gary J Ferland, RL Porter, PAM Van Hoof, RJR Williams, NP Abel, ML Lykins, Gargi Shaw, William J Henney, and PC Stancil. The 2013 release of cloudy. *Revista mexicana de astronomía y astrofísica*, 49(1):137–163, 2013.

[36] Alisa K. Galishnikova, Matthew W. Kunz, and Alexander A. Schekochihin. Tearing instability and current-sheet disruption in the turbulent dynamo, October 2022. arXiv:2201.07757 [astro-ph, physics:physics].

[37] Enrico Garaldi, Rüdiger Pakmor, and Volker Springel. Magnetogenesis around the first galaxies: the impact of different field seeding processes on galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 502(4):5726–5744, 01 2021.

[38] Thomas A. Gardiner and James M. Stone. An unsplit Godunov method for ideal MHD via constrained transport. *Journal of Computational Physics*, 205(2):509–539, May 2005. Publisher: Academic Press Inc.

[39] Thomas A. Gardiner and James M. Stone. An unsplit Godunov method for ideal MHD via constrained transport in three dimensions. *Journal of Computational Physics*, 227(8):4123–4141, April 2008. Publisher: Academic Press Inc.

[40] Frederick A. Gent, Mordecai-Mark Mac Low, Maarit J. Käpylä, and Nishant K. Singh. Small-scale Dynamo in Supernova-driven Interstellar Turbulence. *The Astrophysical Journal Letters*, 910(2):L15, April 2021.

[41] Philipp Girichidis, Thorsten Naab, Stefanie Walch, Michał Hanasz, Mordecai-Mark Mac Low, Jeremiah P. Ostriker, Andrea Gatto, Thomas Peters, Richard Wünsch, Simon C. O. Glover, Ralf S. Klessen, Paul C. Clark, and Christian Baczynski. LAUNCHING COSMIC-RAY-DRIVEN OUTFLOWS FROM THE MAGNETIZED INTERSTELLAR MEDIUM. *The Astrophysical Journal*, 816(2):L19, January 2016.

[42] Philipp Girichidis, Christoph Pfrommer, Michal Hanasz, and Thorsten Naab. Spectrally resolved cosmic ray hydrodynamics – I. Spectral scheme. *Monthly Notices of the Royal Astronomical Society*, page stz2961, November 2019. arXiv:1909.12840 [astro-ph].

[43] Philipp Girichidis, Christoph Pfrommer, Rüdiger Pakmor, and Volker Springel. Spectrally resolved cosmic rays: II – Momentum-dependent cosmic ray diffusion drives powerful galactic winds. *Monthly Notices of the Royal Astronomical Society*, 510(3):3917–3938, January 2022. arXiv:2109.13250 [astro-ph].

[44] Sergei K. Godunov and I. Bohachevsky. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematičeskij sbornik*, 47(89)(3):271–306, 1959.

[45] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)*, 23(1):5–48, 1991.

[46] Robert J. J. Grand, Facundo A. Gómez, Federico Marinacci, Rüdiger Pakmor, Volker Springel, David J. R. Campbell, Carlos S. Frenk, Adrian Jenkins, and Simon D. M. White. The Auriga Project: the properties and formation mechanisms of disc galaxies across cosmic time. *Monthly Notices of the Royal Astronomical Society*, page stx071, January 2017.

[47] Philipp Grete, Joshua C. Dolence, Jonah M. Miller, Joshua Brown, Ben Ryan, Andrew Gaspar, Forrest Glines, Sriram Swaminarayan, Jonas Lippuner, Clell J. Solomon,

Galen Shipman, Christoph Junghans, Daniel Holladay, James M. Stone, and Luke F. Roberts. Parthenon – a performance portable block-structured adaptive mesh refinement framework. *The International Journal of High Performance Computing Applications*, 37(5):465–486, September 2023. arXiv:2202.12309 [astro-ph].

[48] Philipp Grete, Forrest W. Glines, and Brian W. O'Shea. K-Athena: A Performance Portable Structured Grid Finite Volume Magnetohydrodynamics Code. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):85–97, January 2021.

[49] Asger Grønnow, Thor Tepper-García, Joss Bland-Hawthorn, and Filippo Fraternali. The role of the halo magnetic field on accretion through high-velocity clouds. *Monthly Notices of the Royal Astronomical Society*, 509(4):5756–5770, February 2022.

[50] J.L. Han. Observing Interstellar and Intergalactic Magnetic Fields. *Annual Review of Astronomy and Astrophysics*, 55(1):111–157, August 2017.

[51] M. Hanasz, H. Lesch, T. Naab, A. Gawryszczak, K. Kowalik, and D. Wóltański. COSMIC RAYS CAN DRIVE STRONG OUTFLOWS FROM GAS-RICH HIGH-REDSHIFT DISK GALAXIES. *The Astrophysical Journal*, 777(2):L38, October 2013.

[52] Michał Hanasz, Dominik Wóltański, and Kacper Kowalik. GLOBAL GALACTIC DYNAMO DRIVEN BY COSMIC RAYS AND EXPLODING MAGNETIZED STARS. *The Astrophysical Journal*, 706(1):L155–L159, November 2009.

[53] Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. How do scientists develop and use scientific software? In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pages 1–8, 2009.

[54] Amiram Harten, Peter D Lax, and Bram van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1):35–61, 1983.

[55] L. Hatton. *The T-experiments: errors in scientific software*, pages 12–31. Springer US, Boston, MA, 1997.

[56] L. Hatton and A. Roberts. How accurate is scientific software? *IEEE Transactions on Software Engineering*, 20(10):785–797, 1994.

[57] Fernando Hidalgo-Pineda, Ryan Jeffrey Farber, and Max Gronke. Better Together: The Complex Interplay Between Radiative Cooling and Magnetic Draping. *Monthly Notices of the Royal Astronomical Society*, October 2023.

[58] John K Holmen, Philipp Grete, and Veronica G Melesse Vergara. Early Experiences on the OLCF Frontier System with AthenaPK and Parthenon-Hydro. In *CUG Conference Proceedings*, Helsinki, Finland, May 2023.

[59] Philip F. Hopkins. A new class of accurate, mesh-free hydrodynamic simulation methods. *Monthly Notices of the Royal Astronomical Society*, 450(1):53–110, June 2015.

[60] Philip F Hopkins, T K Chan, Shea Garrison-Kimmel, Suoqing Ji, Kung-Yi Su, Cameron B Hummels, Dušan Kereš, Eliot Quataert, and Claude-André Faucher-Giguère. But what about...: cosmic rays, magnetic fields, conduction, and viscosity in galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 492(3):3465–3498, March 2020.

[61] Victoria M. Kaspi and Andrei M. Beloborodov. Magnetars. *Annual Review of Astronomy and Astrophysics*, 55(1):261–301, 2017.

[62] Philipp Kempski, Drummond B Fielding, Eliot Quataert, Alisa K Galishnikova, Matthew W Kunz, Alexander A Philippov, and Bart Ripperda. Cosmic ray transport in large-amplitude turbulence with small-scale field reversals. *Monthly Notices of the Royal Astronomical Society*, 525(4):4985–4998, 09 2023.

[63] Fatima Khan, Muhammad Pasha, and Shahid Masud. Advancements in microprocessor architecture for ubiquitous ai—an overview on history, evolution, and upcoming challenges in ai implementation. *Micromachines*, 12:665, 06 2021.

[64] Chang-Goo Kim and Eve C Ostriker. Momentum injection by supernovae in the interstellar medium. *The Astrophysical Journal*, 802(2):99, 2015.

[65] Chang-Goo Kim and Eve C. Ostriker. VERTICAL EQUILIBRIUM, ENERGETICS, AND STAR FORMATION RATES IN MAGNETIZED GALACTIC DISKS REGULATED BY MOMENTUM FEEDBACK FROM SUPERNOVAE. *The Astrophysical Journal*, 815(1):67, December 2015.

[66] Manish Kumar. Dynamic power dissipation analysis in cmos vlsi circuit design with scaling down in technology. *Journal of Active and Passive Electronic Devices*, pages 55–61, 10 2020.

[67] Stanley Letovsky. Cognitive processes in program comprehension. *Journal of Systems and Software*, 7(4):325–339, 1987.

[68] Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[69] M. T. P. Liska, K. Chatterjee, D. Issa, D. Yoon, N. Kaaz, A. Tchekhovskoy, D. Van Eijnatten, G. Musoke, C. Hesp, V. Rohoza, S. Markoff, A. Ingram, and M. Van Der Klis. H-AMR: A New GPU-accelerated GRMHD Code for Exascale Computing with 3D Adaptive Mesh Refinement and Local Adaptive Time Stepping. *The Astrophysical Journal Supplement Series*, 263(2):26, December 2022.

[70] Federico Marinacci and Mark Vogelsberger. Effects of simulated cosmological magnetic fields on the galaxy population. *Monthly Notices of the Royal Astronomical Society: Letters*, 456(1):L69–L73, 11 2015.

[71] Sergio Martin-Alvarez, Julien Devriendt, Adrianne Slyz, and Romain Teyssier. A three-phase amplification of the cosmic magnetic field in galaxies. *Monthly Notices of the Royal Astronomical Society*, 479(3):3343–3365, September 2018.

[72] Sergio Martin-Alvarez, Adrianne Slyz, Julien Devriendt, and Carlos Gómez-Guijarro. How primordial magnetic fields shrink galaxies. *Monthly Notices of the Royal Astronomical Society*, 495(4):4475–4495, July 2020.

[73] Andrea Mignone and Petros Tzeferacos. A second-order unsplit Godunov scheme for cell-centered MHD: The CTU-GLM scheme. *Journal of Computational Physics*, 229(6):2117–2138, March 2010.

[74] Francesco Miniati and Daniel F. Martin. Constrained-transport magnetohydrodynamics with adaptive mesh refinement in CHARM. *Astrophysical Journal, Supplement Series*, 195(1), July 2011.

[75] Takahiro Miyoshi and Kanya Kusano. A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics. *Journal of Computational Physics*, 208(1):315–344, September 2005. Publisher: Academic Press Inc.

[76] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, Serge Torres, et al. *Handbook of floating-point arithmetic*. Springer, 2018.

[77] Thorsten Naab and Jeremiah P. Ostriker. Theoretical Challenges in Galaxy Formation. *Annual Review of Astronomy and Astrophysics*, 55:59–109, December 2017.

[78] Valery M. Nakariakov and Dmitrii Y. Kolotkov. Magnetohydrodynamic waves in the solar corona. *Annual Review of Astronomy and Astrophysics*, 58(1):441–481, 2020.

[79] Evangelia Ntormousi, Konstantinos Tassis, Fabio Del Sordo, Francesca Fragkoudi, and Rüdiger Pakmor. A dynamo amplifying the magnetic field of a Milky-Way-like galaxy. *Astronomy & Astrophysics*, 641:A165, September 2020.

[80] A. Nuñez-Castiñeyra, I. A. Grenier, F. Bournaud, Y. Dubois, F. R. Kamal Youssef, and P. Hennebelle. Cosmic-ray diffusion and the multi-phase interstellar medium in a dwarf galaxy. I. Large-scale properties and γ-ray luminosities, May 2022. arXiv:2205.08163 [astro-ph].

[81] S. A. Orszag and C. M. Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, 90:129–143, January 1979.

[82] R. Pakmor, C. Pfrommer, C. M. Simpson, and V. Springel. GALACTIC WINDS DRIVEN BY ISOTROPIC AND ANISOTROPIC COSMIC-RAY DIFFUSION IN DISK GALAXIES. *The Astrophysical Journal*, 824(2):L30, June 2016.

[83] Rüdiger Pakmor, Facundo A. Gómez, Robert J. J. Grand, Federico Marinacci, Christine M. Simpson, Volker Springel, David J. R. Campbell, Carlos S. Frenk, Thomas Guillet, Christoph Pfrommer, and Simon D. M. White. Magnetic field formation in the Milky Way like disc galaxies of the Auriga project. *Monthly Notices of the Royal Astronomical Society*, 469(3):3185–3199, August 2017.

[84] Rüdiger Pakmor and Volker Springel. Simulations of magnetic fields in isolated disc galaxies. *Monthly Notices of the Royal Astronomical Society*, 432(1):176–193, June 2013.

[85] Rüdiger Pakmor, Freeke van de Voort, Rebekka Bieri, Facundo A Gómez, Robert J J Grand, Thomas Guillet, Federico Marinacci, Christoph Pfrommer, Christine M Simpson, and Volker Springel. Magnetizing the circumgalactic medium of disc galaxies. *Monthly Notices of the Royal Astronomical Society*, 498(3):3125–3137, September 2020.

[86] Viraj Pandya, Drummond B. Fielding, Greg L. Bryan, Christopher Carr, Rachel S. Somerville, Jonathan Stern, Claude-André Faucher-Giguère, Zachary Hafen, Daniel Anglés-Alcázar, and John C. Forbes. A unified model for the coevolution of galaxies and their circumgalactic medium: The relative roles of turbulence and atomic cooling physics. *The Astrophysical Journal*, 956(2):118, oct 2023.

[87] Vladimir I. Pariev and Stirling A. Colgate. A Magnetic Alpha-Omega Dynamo in Active Galactic Nuclei Disks: I. The Hydrodynamics of Star-Disk Collisions and Keplerian Flow. *The Astrophysical Journal*, 658(1):114–128, March 2007. arXiv: astro-ph/0611139.

[88] Vladimir I. Pariev, Stirling A. Colgate, and John M. Finn. A Magnetic Alpha-Omega Dynamo in Active Galactic Nuclei Disks: II. Magnetic Field Generation, Theories and Simulations. *The Astrophysical Journal*, 658(1):129–160, March 2007. arXiv: astro-ph/0611188.

[89] Christoph Pfrommer, Rüdiger Pakmor, Christine M. Simpson, and Volker Springel. Simulating Gamma-Ray Emission in Star-forming Galaxies. *The Astrophysical Journal*, 847(2):L13, September 2017.

[90] A. Philippov and M. Kramer. Pulsar magnetospheres and their radiation. *Annual Review of Astronomy and Astrophysics*, 60(1):495–558, 2022.

[91] Kenneth G. Powell. *An Approximate Riemann Solver for Magnetohydrodynamics*, pages 570–583. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[92] Prakash Prabhu, Thomas B. Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick P. Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David I. August. A survey of the practice of computational science. In *State of the Practice Reports*, SC '11, New York, NY, USA, 2011. Association for Computing Machinery.

[93] Rahul Ramesh, Dylan Nelson, Drummond Fielding, and Marcus Brüggen. Zooming in on the circumgalactic medium with gible: the topology and draping of magnetic fields around cold clouds, 2024.

[94] Philip L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

[95] Dongsu Ryu and T W Jones. Numerical Magnetohydrodynamics in Astrophysics: Algorithms and Tests for One-Dimensional Flow. *The Astrophysical Journal*, 422:228–258, March 1995.

[96] Dongsu Ryu, TW Jones, and Adam Frank. Numerical magnetohydrodynamics in astrophysics: Algorithm and tests for multidimensional flow. *Astrophysical Journal*, 452(2):785–796, 1995.

[97] Hsi-Yu Schive, John A ZuHone, Nathan J Goldbaum, Matthew J Turk, Massimo Gaspari, and Chin-Yu Cheng. gamer-2: a GPU-accelerated adaptive mesh refinement code – accuracy, performance, and scalability. *Monthly Notices of the Royal Astronomical Society*, 481(4):4815–4840, December 2018.

[98] Evan E. Schneider and Brant E. Robertson. Cholla: A new massively parallel hydrodynamics code for astrophysical simulation. *Astrophysical Journal, Supplement Series*, 217(2):24, 2015. Publisher: IOP Publishing.

[99] Evan E. Schneider and Brant E. Robertson. Introducing CGOLS: The cholla galactic outflow simulation suite. *The Astrophysical Journal*, 860(2), March 2018. Publisher: arXiv.

[100] Swapnil Shankar, Philipp Mösta, Steven R. Brandt, Roland Haas, Erik Schnetter, and Yannick de Graaf. GRaM-X: A new GPU-accelerated dynamical spacetime GRMHD code for Exascale computing with the Einstein Toolkit, November 2022. arXiv:2210.17509 [astro-ph, physics:gr-qc].

[101] Min-Su Shin, James M Stone, and Gregory F Snyder. THE MAGNETOHYDRODYNAMICS OF SHOCK-CLOUD INTERACTION IN THREE DIMENSIONS. *APJ*, 680:336–348, 2008.

[102] Christine M. Simpson, Rüdiger Pakmor, Federico Marinacci, Christoph Pfrommer, Volker Springel, Simon C. O. Glover, Paul C. Clark, and Rowan J. Smith. THE ROLE OF COSMIC-RAY PRESSURE IN ACCELERATING GALACTIC OUTFLOWS. *The Astrophysical Journal*, 827(2):L29, August 2016.

[103] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31, April 1978.

[104] Volker Springel, Rüdiger Pakmor, and Rainer Weinberger. AREPO: Cosmological magnetohydrodynamical moving-mesh simulation code. Astrophysics Source Code Library, record ascl:1909.010, September 2019.

[105] James M. Stone and Thomas Gardiner. A simple unsplit Godunov method for multidimensional MHD. *New Astronomy*, 14(2):139–148, 2009.

[106] James M Stone, Thomas A Gardiner, Peter Teuben, John F Hawley, and Jacob B Simon. ATHENA: A NEW CODE FOR ASTROPHYSICAL MHD. *The Astrophysical Journal Supplement Series*, 178:137–177, 2008.

[107] James M. Stone, Kengo Tomida, Christopher J. White, and Kyle G. Felker. The Athena++ Adaptive Mesh Refinement Framework: Design and Magnetohydrodynamic Solvers. *The Astrophysical Journal Supplement Series*, 249(1):4, June 2020.

[108] Romain Teyssier. Grid-based hydrodynamics in astrophysical fluid flows. *Annual Review of Astronomy and Astrophysics*, 53(1):325–364, 2015.

[109] Eleuteruio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics Third Edition*. Springer, 2009.

[110] Manuel Torrilhon. Locally divergence-preserving upwind finite volume schemes for magnetohydrodynamic equations. *SIAM Journal on Scientific Computing*, 26(4):1166–1191, 2005.

[111] Gábor Tóth. A general code for modeling mhd flows on parallel computers: Versatile advection code. In Yutaka Uchida, Takeo Kosugi, and Hugh S. Hudson, editors, *Magnetodynamic Phenomena in the Solar Atmosphere: Prototypes of Stellar Magnetic Activity*, pages 471–472. Springer Netherlands, Dordrecht, 1996.

[112] J. Trujillo Bueno and T. del Pino Alemán. Magnetic field diagnostics in the solar upper atmosphere. *Annual Review of Astronomy and Astrophysics*, 60(1):415–453, 2022.

[113] Jason Tumlinson, Molly S. Peeples, and Jessica K. Werk. The Circumgalactic Medium. *Annual Review of Astronomy and Astrophysics*, 55:389–432, August 2017. Publisher: Annual Reviews Inc.

[114] Freeke van de Voort, Rebekka Bieri, Rüdiger Pakmor, Facundo A. Gómez, Robert J. J. Grand, and Federico Marinacci. The effect of magnetic fields on properties of the circumgalactic medium. *Monthly Notices of the Royal Astronomical Society*, 501(4):4888–4902, January 2021. arXiv:2008.07537 [astro-ph].

[115] Bram Van Leer. Upwind and High-Resolution Methods for Compressible Flow: From Donor Cell to Residual-Distribution Schemes. In *16th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, April 2006. American Institute of Aeronautics and Astronautics.

[116] Maria Werhahn, Philipp Girichidis, Christoph Pfrommer, and Joseph Whittingham. Gamma-ray emission from spectrally resolved cosmic rays in galaxies. *Monthly Notices of the Royal Astronomical Society*, 525(3):4437–4455, September 2023. arXiv:2301.04163 [astro-ph].

[117] Maria Werhahn, Christoph Pfrommer, Philipp Girichidis, and Georg Winner. Cosmic rays and non-thermal emission in simulated galaxies – II. γ-ray maps, spectra, and the far-infrared–γ-ray relation. *Monthly Notices of the Royal Astronomical Society*, 505(3):3295–3313, June 2021.

[118] Benjamin D. Wibking and Mark R. Krumholz. The global structure of magnetic fields and gas in simulated Milky Way-analogue galaxies. *arXiv:2105.04136 [astro-ph]*, May 2021. arXiv: 2105.04136.

[119] Greg Wilson, D A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven H D Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, Ben Waugh, Ethan P White, and Paul Wilson. Best Practices for Scientific Computing. *PLOS Biology*, 12(1):7, 2014.

[120] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. Good enough practices in scientific computing. *PLOS Computational Biology*, 13(6):e1005510, June 2017.

[121] H.-Y. K. Yang, M. Ruszkowski, P. M. Ricker, E. Zweibel, and D. Lee. THE FERMI BUBBLES: SUPERSONIC ACTIVE GALACTIC NUCLEUS JETS WITH ANISOTROPIC COSMIC-RAY DIFFUSION. *The Astrophysical Journal*, 761(2):185, December 2012.

[122] Kotaro Yoshida, Shuichi Matsukiyo, Kesuke Shimokawa, Haruichi Washimi, and Tohru Hada. Trajectory Analysis of Galactic Cosmic Rays Invading into the Heliosphere. *The Astrophysical Journal*, 916(1):29, July 2021.

[123] M. Zingale, A. S. Almgren, M. Barrios Sazo, J. B. Bell, K. Eiden, A. Harpole, M. P. Katz, A. J. Nonaka, D. E. Willcox, and W. Zhang. The Castro AMR Simulation Code: Current and Future Developments. *Journal of Physics: Conference Series*, 1623(1):012021, September 2020.