**grim: A Flexible, Conservative Scheme for Relativistic Fluid Theories**

Mani Chandra¹, François Foucart²,⁴, and Charles F. Gammie¹,³

¹Department of Astronomy, University of Illinois, 1110 West Green Street, Urbana, IL, 61801, USA; manic@illinois.edu, gammie@illinois.edu

²Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA; vfoucart@lbl.gov

³Department of Astronomy, University of Illinois, 1002 West Green Street, Urbana, IL, 61801, USA

Received 2016 August 31; revised 2017 January 26; accepted 2017 February 1; published 2017 March 7

**Abstract**

Hot, diffuse, relativistic plasmas such as sub-Eddington black-hole accretion flows are expected to be collisionless, yet are commonly modeled as a fluid using ideal general relativistic magnetohydrodynamics (GRMHD). Dissipative effects such as heat conduction and viscosity can be important in a collisionless plasma and will potentially alter the dynamics and radiative properties of the flow from that in ideal fluid models; we refer to models that include these processes as Extended GRMHD. Here we describe a new conservative code, grim, that enables all of the above and additional physics to be efficiently incorporated. grim combines time evolution and primitive variable inversion needed for conservative schemes into a single step using an algorithm that only requires the residuals of the governing equations as inputs. This algorithm enables the code to be physics agnostic as well as flexibility regarding time-stepping schemes. grim runs on CPUs, as well as on GPUs, using the same code. We formulate a performance model and use it to show that our implementation runs optimally on both architectures. grim correctly captures classical GRMHD test problems as well as a new suite of linear and nonlinear test problems with anisotropic conduction and viscosity in special and general relativity. As tests and example applications, we resolve the shock substructure due to the presence of dissipation, and report on relativistic versions of the magneto-thermal instability and heat flux driven buoyancy instability, which arise due to anisotropic heat conduction, and of the firehose instability, which occurs due to anisotropic pressure (i.e., viscosity). Finally, we show an example integration of an accretion flow around a Kerr black hole, using Extended GRMHD.

**Key words:** accretion, accretion disks – magnetohydrodynamics (MHD) – plasmas – relativistic processes

1. Introduction

The fluid description of a plasma using the ideal general relativistic magnetohydrodynamics (GRMHD) equations is a workhorse in theoretical high energy astrophysics. The codes that solve these equations have been successfully applied in studies of various processes of interest such as jet formation and accretion onto compact objects. Many important results have emerged from numerical solutions of the ideal GRMHD equations. A few examples are the validation that the Blandford & Znajek (1977) mechanism occurs naturally in a global MHD model (McKinney & Gammie 2004), the discovery of magnetically chocked accretion flows (Tchekhovskoy et al. 2011; McKinney et al. 2012), and simulated observations of Sgr A* (Mościbrodzka et al. 2009).

However, the ideal GRMHD model is readily justified only when the Knudsen number \( \frac{l_{\text{mfp}}}{l_{\text{system}}} \ll 1 \), where \( l_{\text{mfp}} \) is the mean-free path, and \( l_{\text{system}} \) is the characteristic length scale of the system, and when the ratio of the timescales \( \tau_C/\tau_D \ll 1 \), where \( \tau_C \) is the two-body Coulomb scattering timescale, and \( \tau_D \) is the dynamical timescale in the system. In other words, the ideal GRMHD model assumes that the plasma is locally in equilibrium. This leads to a simple set of conservation laws for mass and momentum and all that is required to complete the system is a prescription for the pressure, which is usually approximated by a Gamma-law equation of state. While this simplicity is appealing, systems such as low-luminosity black holes, which accrete through a radiatively inefficient accretion flow (RIAF) are in the \( \text{Kn} \gg 1 \) regime.

In a RIAF, the synchrotron cooling timescales are much longer than the dynamical timescale. This leads to the accreting plasma becoming virially hot because the gravitational potential energy is stored as internal energy, with \( T \sim R^{-1} \), where \( T \) is the temperature of the plasma, and \( R \) is the radius from the black hole. The disk is then geometrically thick and optically thin (Yuan & Narayan 2014), and the Coulomb mean-free-paths between all the constituent particles (ion–ion, ion–electron, electron–electron) (all of which scale as \( \sim T^2 \)) are much larger than the typical system scale \( GM/c^2 \) (Mahadevan & Quataert 1997). Thus, it is not evident that ideal GRMHD is applicable.

Despite the divergence of the Knudsen number, and the collisional timescale, there are indeed small parameters that can be exploited to recover an effective hydrodynamic description. In the presence of a sufficiently strong magnetic field, the following conditions can apply: \( l_{\text{gyro}}/l_{\text{system}} \ll 1 \), and \( l_{\text{gyro}}/\tau_{\text{gyro}} \ll 1 \), where \( l_{\text{gyro}} \) is the gyroradius and \( \tau_{\text{gyro}} \) is the gyropoint. These apply in most astrophysical systems. For example, in Sgr A*, Faraday rotation measurements and observed synchrotron radiation indicate a magnetic field strength of \( \sim 100 \) Gauss and a number density of \( \sim 10^7 \text{ cm}^{-3} \), implying \( l_{\text{gyro}}/l_{\text{system}} \sim 10^{-5} \) and \( l_{\text{gyro}}/\tau_{\text{gyro}} \sim 10^{-8} \). Thus, particles are constrained to move along field lines. In the presence of weak collisionality, perhaps provided by wave-particle scattering, this leads to a set of fluid-like equations with anisotropic transport along the magnetic field lines.

Dissipative relativistic fluid theories should be hyperbolic, causal, and stable. Early theories by Eckart (1940) and Landau–Lifshitz do not satisfy these requirements, whereas these are conditionally satisfied by the Israel & Stewart (1979)...
theory of dissipative hydrodynamics (Hiscock & Lindblom 1983, 1985, 1988a, 1988b, p. 181). Chandra et al. (2015) adapted the Israel & Stewart (1979) theory for isotropic conduction and viscosity, taking into account the symmetries imposed on the distribution function of a plasma in the presence of a magnetic field to derive a one-fluid model of a plasma that incorporates anisotropic thermal conduction and viscosity. The conduction is driven by temperature gradients along field lines and the viscosity due to a shear flow projected onto the field lines. The model, referred to as extended magnetohydrodynamics (EMHD), is valid up to second-order deviations from equilibrium and is applicable to weakly collisional flows. We review the equations of the model in Section 4 and encourage the interested reader to look at Chandra et al. (2015) for the derivation and the limits of the model within which it satisfies the above mentioned constraints. In this paper, we derive a variety of analytic and semianalytic solutions, described in Section 8, to develop intuition about the EMHD model, and to serve in a test suite for the numerical implementation of EMHD and similar models.

The methods used to integrate the equations of relativistic MHD are similar to those used in non-relativistic MHD, namely, shock capturing conservative schemes using the finite volume method. In particular, the approximate Riemann solvers used to compute the numerical fluxes at cell interfaces, and the various methods available to evolve the magnetic field under the constraint $\nabla \cdot B = 0$ are similar for relativistic and non-relativistic MHD. One of the main complications in relativistic MHD is the mathematical relation between the evolved variables and physical variables. Consider special-relativistic ideal hydrodynamics, where the physical variables to be solved for, referred to as primitive variables, are the restmass energy density $\rho$, the internal energy $e$, and the spatial components of the four-velocity $u^i$. The variables are evolved using the continuity equation $\partial_\nu (\rho u^\nu) = 0$, and the energy and momentum conservation equations given by $\partial_\nu T^{\nu\nu} = 0$, where $T^{\nu\nu} = (\rho + P) u^\nu u^\nu + P n^{\nu\nu}$ is the perfect fluid stress tensor, $m$ is the particle mass, $\eta^{\nu\mu} = \text{diag}(-1, 1, 1, 1)$ is the flat space metric, and $P$ is the pressure, approximated here by a gamma-law equation of state, $P = (\gamma - 1) u^0$. Conservative schemes time-step the conserved variables, $U = (\rho u^0, T_{00})$, from $U^n$ to $U^{n+1}$, where the superscripts $n, n + 1$ indicate the discretized time levels. To recover the primitive variables $u^{n+1}$ and $(u^i)^{n+1}$ at the new time step from $U^{n+1}$ requires the solution to a set of nonlinear equations and is a multivariate nonlinear root finding problem (though for hydrodynamics it can be reduced to a univariate nonlinear problem). This is unlike non-relativistic fluid dynamics, where this recovery step is algebraic.

Many schemes have been proposed for the recovery of primitive variables from conserved variables in relativistic hydrodynamics (Noble et al. 2006). However, the introduction of new physics, as in the EMHD model, voids the earlier algorithms, which are specialized to ideal MHD. The model has equations governing the dissipative quantities $q$, the heat flux along the magnetic field lines, and $\Delta P$, the pressure anisotropy, which are of the form $\partial_\nu q = \tilde{b}^\nu \partial_\nu T + \tilde{b}_\nu \nu\nu \partial_\nu u_\nu$ and $\partial_\nu \Delta P = \tilde{b}^\nu \tilde{b}_\nu \partial_\nu u_\nu$, where $T$ is the temperature and $\tilde{b}^\nu$ is the unit vector along the direction of the magnetic field. The difficulty is that the equations for $q$ and $\Delta P$ are sourced by spatio-temporal derivatives and not just spatial derivatives. The values of $q^{n+1}$ and $\Delta P^{n+1}$ depend on the values of $T^{n+1}$ and $u^{n+1}_\nu$, but these in turn need to be recovered from the conserved quantities $U^{n+1}$. Now, the stress tensor has dissipative contributions of the form $T^{\nu\nu} \sim q (\tilde{b}^\nu u^\nu + \tilde{b}^\nu u^\nu) + \Delta P \tilde{b}^\nu \tilde{b}^\nu$ and so $U^{n+1}$ itself depends on $q^{n+1}$ and $\Delta P^{n+1}$. Thus, the time evolution of all thermodynamic quantities are nonlinearly intertwined with primitive variable recovery.

grim recasts the entire time-stepping procedure as a coupled multivariate nonlinear root finding problem. Consider, as a simple example, the following system of 1D wave-equations:

$$\partial_t u_1 + c \partial_x u_1 = 0 \quad (1)$$
$$\partial_t u_2 + c \partial_x u_2 = 0 \quad (2)$$

for the variables $u_1(x, t)$ and $u_2(x, t)$. Now, performing an explicit first order spatio-temporal discretization, we have $(u_1^{n+1} - u_1^n) / \Delta t + c (u_1^{n+1} - u_1^n) / \Delta x = 0$ (assuming $c > 0$), where the index $i$ denotes a grid zone and the index $n$ denotes a time level. Here, both $u_1^{n+1}$ and $u_2^{n+1}$ can be solved for algebraically, $u_1^{n+1} = u_1^n - c \Delta t / \Delta x (u_1^{n+1} - u_1^n)$. In grim, we find instead the values of $u_1^{n+1}$ and $u_2^{n+1}$ that satisfy

$$f(u_1^{n+1}, u_2^{n+1}) = 0 \quad (3)$$

where $f(u_1^{n+1}, u_2^{n+1})$ are the residuals, and represent the governing equations in their discretized form. This system of equations (which, in general, are nonlinearly coupled) is now solved using an iterative algorithm until $|f(u_1^{n+1}, u_2^{n+1})| < \epsilon$, where $|.|$ is a suitable norm and $\epsilon$ is a chosen tolerance. The algorithm requires, as sole input, the residuals $f(\cdot)$, which are the discretized form of the governing equations. The algorithm is independent of the physics that constitutes the discretized equations $f(\cdot)$ and is therefore independent of the underlying physical model. It works with ideal MHD, EMHD, and possible extensions of the EMHD model. Thus, the abstraction of the numerical solution to a set of PDEs as a nonlinear root finding problem allows for flexibility regarding the governing equations, as well as time-stepping schemes as we shall show in later sections.

We begin in Section 2 by describing the numerical discretization of a set of hyperbolic PDEs to $O(\Delta x^2, \Delta t^2)$ using the finite volume method combined with a semi-implicit time-stepping scheme. We then proceed in Section 3 to recast the time stepping of the discrete system as a nonlinear multivariate root finding problem and describe how the roots are obtained using a residual-based algorithm. We then apply this technique to the EMHD model in Section 4, along with a review of the governing equations. We detail the implementation of all of the above in Section 5, and describe the techniques, and libraries we use that enable us to use either CPUs, or GPUs. We then report various performance and scaling data in Section 6. In order to understand the performance numbers, we formulate a performance model in Section 7, and use it to show that our implementation is optimal on both CPUs, and GPUs. We have developed an extensive test suite for the EMHD model, which we present in Section 8, and validate grim using this test suite, thus demonstrating its utility in exploring the

---

5 General Relativistic Implicit Magnetohydrodynamics: http://github.com/afd-illinois/grim. Commit hash used in this paper: 76bec371.
solution space of this model. In Section 9, we show example applications of grim; buoyancy instabilities that occur in weakly collisional plasmas, and accretion onto supermassive black holes. Finally, in Section 10, we conclude.

2. Finite Volume Method

grim uses the finite volume method to solve hyperbolic partial differential equations in their conservative form

$$\partial_t U + \partial_j F^j = S$$

(4)

where $U$ is the vector of conserved quantities, $F^j$ are fluxes, and $S$ are sources. We break down the full scheme into (Section 2.1) the domain discretization, (Section 2.2) the integral form of the differential equations, (Section 2.3) the time-stepping scheme, and (Section 2.4) the spatial discretization.

2.1. Grid Generation

We are primarily interested in solving (4) in simple rectangular and spherical geometries. To discretize these domains, we work in coordinates where the boundaries of the domains are aligned with the coordinate axes. For example, Cartesian coordinates $x^i = \{x, y, z\}$ for rectangular domains, and spherical polar coordinates $x^i = \{r, \theta, \phi\}$ for spherical geometries. Then, given the extent of the domain in these coordinates $[x^{i,\text{start}}, x^{i,\text{end}}]$, a grid with $N_1 \times N_2 \times N_3$ zones is generated by decomposing the spatial domain into zones with dimensions $dx^1 \times dx^2 \times dx^3$, where $dx^i = (x^{i,\text{end}} - x^{i,\text{start}})/N^i$, for $i = 1, 2, 3$. This results in a uniform mesh in each coordinate.

If a physical problem requires the concentration of grid zones in a specific region, we construct a smooth curvilinear non-uniform grid using a coordinate transformation, as is done in the harm code (Gammie et al. 2003). First, a uniform grid is generated in a different set of coordinates $X^i$, and then transformed to the $x^i$ coordinates using $x^i \equiv x^i(X^i)$. The grid zones in $X^i$ all have equal dimensions $dX^1 \times dX^2 \times dX^3$, where $dX^i = (X^{i,\text{end}} - X^{i,\text{start}})/N^i$, and $[X^{i,\text{start}}, X^{i,\text{end}}]$ is the extent of the domain in the new coordinates. This corresponds to a grid spacing of $dx^i = L^i_j dx^j$ in $x^i$, where $L^i_j = \partial x^i / \partial X^j$ is the transformation matrix. Depending on the form of $x^i(X^i)$, a non-uniform grid is generated in the $x^i$ coordinates.

Below, we illustrate the grid generation for a domain enclosed by two spherical shells (see Figure 1). We concentrate the grid zones near the inner radius $r_{in}$ using a log($r$) grid, and near the midplane $\theta = \pi/2$, with an adjustable parameter $h = (0, 1]$. As $h \to 0$, there is greater concentration of the zones near the midplane.

$$x^1 \equiv r = \exp(X^1)$$

$$x^2 \equiv \theta = \pi X^2 + \left(1 - \frac{h}{2}\right) \sin(2\pi X^2)$$

$$x^3 \equiv \phi = X^3$$

$$L^i_j = \begin{pmatrix}
\exp(X^i) & 0 & 0 \\
0 & \pi(1 + (1 - h) \cos(2\pi X^2)) & 0 \\
0 & 0 & 1
\end{pmatrix}$$

(8)

The boundaries of the domain in $x^i = \{r, \theta, \phi\}$ are $[r_{in}, r_{out}] \times [0, \pi] \times [0, 2\pi]$, which correspond to $[\log(r_{in}), \log(r_{out})] \times [0, 1] \times [0, 1]$ in $X^i = \{X^1, X^2, X^3\}$ coordinates.

2.2. Integral Form of the Differential Equations

We now setup the finite volume formulation in the $(X^1, X^2, X^3)$ coordinate system. Multiplying (4) by the area of the control volume (see figure 2) $\Delta v = dx^1 dx^2 dx^3$, we get

$$\partial_t \int U \Delta v + \int dX^i \partial_t \left( \int F^i dx^2 dx^3 \right)$$

$$+ \int dX^2 \partial_2 \left( \int F^2 dx^1 dx^3 \right) + \cdots = \int S \Delta v.$$  

(9)

Rewriting the above in terms of cell-averages $\bar{U} = \int U \Delta v / \int \Delta v$, $\bar{S} = \int S \Delta v / \int \Delta v$ and the face-averages $\bar{F}^i = \int F^i dx^2 dx^3 / \int dx^2 dx^3$, $\bar{F}^2 = \int F^2 dx^1 dx^3 / \int dx^1 dx^3$, and

Figure 1. Generation of a spherical grid with midplane grid refinement, with the refinement parameter $h = 0.3$ in (6). The left side shows the grid in the computational coordinates ($X^1, X^2$) that grim works in, and the right side shows the grid in Cartesian coordinates.
\[ F^3 \equiv \int F^3 d\mathbf{x} d\mathbf{x} / \int d\mathbf{x} d\mathbf{x}^2 \]

\[ \partial_t \bar{U} + \frac{\bar{F}^\text{right} - \bar{F}^\text{left}}{\Delta X^1} + \frac{\bar{F}^\text{top} - \bar{F}^\text{bottom}}{\Delta X^2} + \frac{F^3_{\text{front}} - F^3_{\text{back}}}{\Delta X^3} = \bar{S} \]

(10)

where we have replaced \( \int d\mathbf{x}^i \partial_i() \) in (9) by the surface integral of \( F^1 \) on the right and left surfaces of the control volume (see Figure 2), and \( \int d\mathbf{x}^2 \partial_2() \), \( \int d\mathbf{x}^3 \partial_3() \) have been replaced by surface integrals of \( F^2 \) on the top and bottom surfaces, and similarly for \( F^3 \) on the front and back surfaces respectively. The above Equation (10) are an exact integral reformulation of the differential Equation (4) over the control volume. Multiplying (10) by \( d\mathbf{t} \) and performing the integration over a discrete time interval \( \Delta t \),

\[ U_n + \frac{1}{\Delta X^1} \int d\mathbf{x}^1 \partial_1() \bigg|_{U_n}^{U_{n+1}} - \int d\mathbf{x}^1 \partial_1() \bigg|_{U_{n+1}}^{U_n} = \frac{1}{\Delta t} \int d\mathbf{S} \]

(11)

where the index \( n \) indicates the discrete time level. Equation (11) are evolution equations for the zone-averaged conserved variables \( U_{n+1} \), which are in turn (nonlinear) functions of the zone-averaged primitive variables \( P_{n+1} \), i.e., \( U_{n+1} \equiv U(P_{n+1}) \).

To proceed, we need to evaluate the spatial integrals \( \int d\mathbf{v} \) and the temporal integrals \( \int d\mathbf{t} \) in (11) using a numerical quadrature to a desired order. We opt for a truncation error of \( O(\Delta t^2) \), \( \Delta X^2 \). The required accuracy can be achieved by evaluating the spatial integrals as \( \int d\mathbf{x}^1() \to \Delta X^1() \), \( \int d\mathbf{x}^2() \to \Delta X^2() \), and \( \int d\mathbf{x}^3() \to \Delta X^3() \), where the spatial integer indices \( i, j, k \) indicate the zone centers in the \( X^1, X^2, X^3 \) directions respectively. The outcome of this quadrature procedure is that the cell-averaged conserved variables \( \bar{U} \), and the cell-averaged source terms \( \bar{S} \) can be replaced by point values \( U_{i,j,k} \) and \( S_{i,j,k} \) at the center of a grid zone and the face-averaged fluxes \( F^1 \) in the \( X^1 \) direction can be replaced by point values at the centers of the right and left faces, \( F^1_{\text{right}} \approx F^1_{i+1/2,j,k} \) and \( F^1_{\text{left}} \approx F^1_{i-1/2,j,k} \) respectively. The substitution for the face-averaged fluxes \( F^2 \) in the \( X^2 \) direction, and \( F^3 \) in the \( X^3 \) direction, by point values follows on similar lines.

### 2.3. Time-stepping Scheme

The temporal integral \( \int d\mathbf{t}() \) for the various terms in (11) is approximated to \( O(\Delta t^2) \) using a two-stage semi-implicit scheme designed to deal with stiff source terms. Depending on the theory being solved for, the source terms can have spatio-temporal derivatives \( S = S(P, \partial_i P, \partial_i \partial_j P) \). We separate these as \( S = S^I(P) + S^E(P) + A'(P)\partial_i P + A'(P)\partial_j P \), where \( S^I(P) \) denote source terms to be treated implicitly (I) or explicitly (E), and \( A'(P) \) are the coefficients of the temporal \( \partial_i P \) and spatial derivative terms \( \partial_j P \) respectively. The spatial derivative terms, when present in the sources, are evaluated using slope limited derivatives on a symmetric stencil (currently the generalized minmod slope using a three-points stencil, though higher-order schemes inspired by the WENO5 (Liu et al. 1994; Jiang & Shu 1996) and PPM (Colella & Woodward 1984) methods are also implemented). The scheme proceeds in two stages.

1. First, we take a half step to go from \( P_n \to P_{n+1/2} \), where the index \( n + 1/2 \) indicates the half time step. The temporal integrals for the fluxes \( \partial_i F^1 \), for the explicit sources \( S^E \), and for the spatial derivative terms in the sources are evaluated explicitly using \( \int d\mathbf{t}() \to (\Delta t/2)(\cdot)_n \), whereas the sources \( S^I \) are treated implicitly using \( \int d\mathbf{t}() \to (\Delta t/2)(\cdot)_{n+1/2} + (\cdot)_n \). This leads to the following discrete form

\[ U(P_{n+1/2}) - U(P_n) + \frac{1}{\Delta t} \int d\mathbf{t} F^1_{\text{right}}(P_n) - F^1_{\text{left}}(P_n) \]

(12)

2. Next, we take a full step from \( P_n \to P_{n+1} \). The temporal integrals for \( \partial_i F^1 \), \( S^E \), and \( A'(P)\partial_i P \) are evaluated using \( \int d\mathbf{t}() \to \Delta t((\cdot)_{n+1/2} + O(\Delta t^2)) \). This is performed using \( P_{n+1/2} \) obtained from the half step. The source terms \( S^I \) are treated implicitly using \( \int d\mathbf{t}() \to \Delta t((\cdot)_{n+1} + (\cdot)_n) + O(\Delta t^2) \)

\[ U(P_{n+1}) - U(P_{n+1/2}) + \frac{1}{\Delta t} \int d\mathbf{t} F^1_{\text{right}}(P_{n+1/2}) - F^1_{\text{left}}(P_{n+1/2}) \]

(13)

This is an unconventional definition of source terms, but it allows us to use a notation that is as closely analogous to non-relativistic fluids as possible.
where (...) denotes flux discretizations in $X^2$ and $X^3$, which we have not written for brevity.

The separation between explicit and implicit sources $S^{l,e}$ is problem-dependent. Stiff source terms are treated implicitly, while computationally expensive source terms can be treated explicitly if desired. For additional flexibility, nonlinear source terms can also use a mixed implicit–explicit approach. For example, the extended MHD algorithm has source terms of the form

$$S(P) = \frac{P - P_0(\partial_t P)}{\tau_R[P]}$$

where $\tau_R$ is a potentially small damping timescale. In this case, it is advantageous to treat $P/\tau_R[P]$ implicitly and $P_0$ explicitly. However, it is also preferable to use a consistent damping timescale $\tau_R[P]$ for all terms. Accordingly, for the half time step, we use

$$\int dtS(P) = \frac{\Delta t}{2} \left( \frac{P_{n+1/2} + P_n}{2\tau_R[P_{n}]} - \frac{P_0(\partial_t P_n)}{\tau_R[P_{n}]} \right),$$

and for the full time step,

$$\int dtS(P) = \Delta t \left( \frac{P_{n+1} + P_n}{2\tau_R[P_{n+1/2}]} - \frac{P_0(\partial_t P_{n+1/2})}{\tau_R[P_{n+1/2}]} \right).$$

This is easily implemented as long as the implicit source terms $S^l$ have access to $P_n$ during the half step and $P_{n+1/2}$ during the full step. In practice, for any system of equations, the user is responsible for providing functions $S^l(P, P^e), S^e(P^e), \ldots$, with $P^e = P_n$ for the half step and $P^e = P_{n+1/2}$ for the full step. The code then assembles the evolution equations from the discretization described in this section.

Evidently, the above system of equations obtained using a semi-implicit temporal discretization requires us to solve a set of nonlinearly coupled equations for $P_{n+1/2}$ and $P_{n+1}$ in the half step (12), and the full step (13) respectively. Furthermore, the presence of time derivatives $(\partial_t P)\partial_t P$ in the source terms implies that we cannot separately time step the conserved variables $U_n \rightarrow U_{n+1} \equiv U(P_{n+1})$, and invert them later to obtain $P_{n+1}$, as is usually the case. The time stepping and the inversion must be done simultaneously. We describe the algorithm to do this in Section 3. However, we note that equations without implicitly coupled source terms are treated explicitly, and do not require the nonlinear solver.

### 2.4. Flux Computation

The computation of the face-centered fluxes $F_{i-1/2}^l \equiv F_i^l(P_{i-1/2}^-; P_{i-1/2}^+) \text{ and } F_{i+1/2}^l \equiv F_i^l(P_{i+1/2}^-; P_{i+1/2}^+)$ requires two stages: (1) reconstruction of the primitive variables from the cell centers $P_{i-1/2}, \ldots, P_{i+1/2}$ to the left $P_{i-1/2,i+1/2}$ and right $P_{i-1/2,i+1/2}$ side of the face centers at $i = -1/2, i + 1/2$, and (2) a Riemann solver to evaluate the fluxes $F_{i+1/2}^l$ given the left $P_{i-1/2,i+1/2}$ and the right states $P_{i+1/2,i+1/2}$.

#### 2.4.1. Reconstruction

The face-centered primitive variables are obtained using a reconstruction operator $R$. The operator takes as input the values of adjacent zone-centered primitive variables to construct a polynomial interpolant to a desired order inside the zone, which is then evaluated at the face centers. We now describe the reconstruction procedure in one dimension, along $X^1$. For brevity, we suppress the $X^2$ and $X^3$ zone indices.

Multidimensional reconstruction proceeds by performing the one-dimensional reconstruction separately in each direction.

For a zone with center $i$, the reconstruction operation $R$ is used in two ways depending on the input order. In the case of a three-point reconstruction stencil, we use $R_i^l = R(P_{i-1}, P_i, P_{i+1})$ to give $F_{i+1/2}^l$, the primitives variables on the left side of the right face of the zone and $R_i^r = R(P_{i+1}, P_i, P_{i-1})$ to give $F_{i-1/2}^l$, the primitive variables at the right side of the left face of the zone. This procedure is repeated for the zone with center $i - 1$ with $R_{i-1}^r$ to obtain $F_{i-1/2}^l$, and for the zone with center $i + 1$ with $R_{i+1}^l$ to obtain $F_{i+1/2}^l$. We now have the states $F_{i-1/2}^l$ needed by the Riemann solver to compute the fluxes $F_{i-1/2}^l \equiv F_{i-1/2}(P_{i-1/2}^-, P_{i+1/2}^+)$, and $F_{i+1/2}^l$ needed to compute $F_{i+1/2}^l = F_{i+1/2}(P_{i+1/2}^-, P_{i+1/2}^+)$.  

#### 2.4.2. Riemann Solver

For generic systems of equations, we have to rely on relatively simple Riemann solvers—at least if we want to avoid numerical computation of the characteristic speeds and eigenvectors of the evolution system. Here, we rely on either the Local Lax Friedrich (LLF) flux, or the HLLE flux (Harten et al. 1983). The LLF and HLLE solvers rely on the knowledge of the fluxes $F_{i}^\pm$ and the conservative variables $U_{i}^\pm$ on the right/left side of face $i$. Both are computed directly from the reconstructed primitive variables $P_{i}^\pm$. For the LLF flux, we also use an estimate of the maximum characteristic speed on face $i$, $c_{max,i} \geq \max(|c_{j,i}^\pm|)$, where $c_{j,i}$ is the $j$th speed on face $i$. The LLF flux is then

$$F_i^\text{LLF} = \frac{F_i^+ - F_i^-}{2} - \frac{\max(0, c_{max,i}^+)}{2}(U_i^+ - U_i^-).$$

Similarly, the HLLE flux relies on estimates of the maximum left-going and right-going characteristic speeds on face $i$, $c_{max,i}^R \geq \max(c_{j,i}^R, 0)$ and $c_{max,i}^L \geq \max(-c_{j,i}^L, 0)$. The HLLE flux is then

$$F_i^\text{HLLE} = \frac{F_i^+ - F_i^-}{2} + \frac{\max(0, c_{max,i}^R)}{2}(U_i^+ - U_i^-) + c_{max,i}^L.$$
by recasting the equations to be solved for, as residuals $R(P)$, where $R$ is the vector of equations, and $P$ are the unknown primitive variables. For example, the residuals for the half step evolution (12) are

$$R(P_{n+1/2}) = \frac{U(P_{n+1/2}) - U(P_{n})}{\Delta t/2} + \nabla_i^r(P_{n+1/2}) - \nabla_i^l(P_{n}) + \ldots$$

$$- S^k(P_{n+1/2}) - \frac{1}{2} (S^l(P_{n+1/2}) + S^l(P_{n}))$$

$$- A^l(P_{n}) \frac{P_{n+1/2} - P_{n}}{\Delta t/2} - A^l(P_{n}) \partial_t P_{n}$$  \hspace{1cm} (19)

Given the residuals as a function of the unknowns $R = R(P)$, the algorithm proceeds by starting with a guess for the unknowns $P_{n+1/2}$ and iterating using

$$P^{k+1} = P^k + \chi^k \delta P^k$$  \hspace{1cm} (20)

for $k = 0, 1, \ldots, k_{\text{max}}$ till $||R(P)|| < \text{tol}$, where $||.||$ is a suitable norm, tol is a desired tolerance, $\delta P^k$ is a linear correction, which we describe in Section 3.1, and $\chi^k \in (0, 1]$ is a line search parameter, which is determined by a quadratic backtracking line search strategy that we describe in Section 3.2. In writing the above, we have suppressed the half step index $n + 1/2$.

3.1. Residual-based Jacobian Computation

The correction $\delta P^k$ at each nonlinear iteration $k$ is obtained by solving the following linear system of equations

$$J(P^k) \delta P^k = - R(P^k)$$  \hspace{1cm} (21)

where the matrix $J(P^k)$ is the $N_{\text{var}} \times N_{\text{var}}$ Jacobian of the system evaluated at $P^k$, and $N_{\text{var}}$ is the number of primitive variables being solved for. The Jacobian itself is assembled numerically from the residual function $R(P)$ to $O(\epsilon)$, where $\epsilon$ is a small differencing parameter. Column $i$ and row $j$ of the numerical Jacobian are computed using

$$J_{ij}(P^k) \approx \frac{R_i(P^k + \epsilon P^j) - R_i(P^j)}{\epsilon},$$  \hspace{1cm} (22)

and the perturbed unknowns $P^j$ are given by

$$P^j = (1 + \epsilon)P^k (1 - \text{small}(P^k)) + \epsilon \cdot \text{small}(P^k)$$  \hspace{1cm} (23)

where

$$\text{small}(P^k) = \begin{cases} 1, & |P^k| < 5 \epsilon \\ 0, & \text{otherwise.} \end{cases}$$

The use of the function small($P^k$) in $P^j$ prevents a division by zero in (21). In the absence of small($P^k$), this occurs when any component of $P^k = 0$, leading to $P^j = 0$.

3.2. Line Search

The traditional Newton algorithm is given by (20), with $\lambda^k = 1$. However, this is not robust and can diverge from the solution. When that happens we backtrack by choosing $\lambda \in (0, 1]$ according to the following strategy.

1. Initialize $\lambda^k = 1$.
2. If $||R(P^k + \chi^k \delta P^k)|| < ||R(P^k)|| (1 - \epsilon_{\text{GR}} \lambda^k)$, accept the new guess $P^{k+1} = P^k + \chi^k \delta P^k$ for the primitive variables and exit the line search. Otherwise, continue to the computation of a new $\lambda^k$. In grIMP, we set the small parameter $\epsilon_{\text{GR}} = 10^{-4}$. If this condition is satisfied, we know that the current guess $P^k + \chi^k \delta P^k$ provides at least some improvement over the previous guess $P^k$.
3. Find the new line search parameter $\lambda_{\text{new}}$ by minimizing the function

$$f(\lambda) = ||R(P^k + \lambda \delta P^k)||^2,$$  \hspace{1cm} (24)

modeling $f$ as a quadratic function of $\lambda$ and using the fact that $df/d\lambda|_{\lambda = 0} = -2f(0)$ (because $\delta P^k$ is the solution of the linear problem at $P^k$). We then have

$$\lambda_{\text{new}}^k = \frac{f(0)}{f(\lambda_{\text{old}}^k) + (2\lambda_{\text{old}}^k - 1)f(0)} \lambda_{\text{old}}^k$$  \hspace{1cm} (25)

We then set $\lambda^k = \lambda_{\text{new}}^k$ and go back to the previous step.

We note that this procedure is performed separately at each point.

4. Extended GRMHD

The EMHD model (Chandra et al. 2015) is a one-fluid model of a plasma consisting of electrons and ions. It considers the following number current vector $N^\nu$ for the ions (set to be the same for electrons) and total (electrons+ions) stress tensor $T^{\mu\nu}$

$$N^\mu = n u^\mu$$  \hspace{1cm} (26)

$$T^{\mu\nu} = \left( \rho + \frac{1}{2} b^2 \right) u^\mu u^\nu + P_g + \frac{1}{2} b^2 h^{\mu\nu} - b^i b^j u^i u^j + q^i u^i + \Pi^{\mu\nu}$$  \hspace{1cm} (27)

where $n$ is the number density of ions, which is equal to the number density of electrons, $\rho = (m_i + m_e)n \approx m_in$ is the total rest-mass energy density, $m_i$ and $m_e$ are the electron and proton rest masses, $u$ is the total internal energy, $P$ is the total pressure approximated by a Gamma-law equation of state $P_g = (\gamma - 1)u$, $u^\mu$ is a four-velocity whose choice here corresponds to an observer comoving with the number current, also known as the Eckart frame, and $b^i$ is a magnetic field four-vector whose components are given by $b^i = B^i w g_{\mu i}$, $b^i = (B^i + b^i u^i)/u^i$, where the magnetic field three-vector $B^i = F^{*i}$, and $F^i$ is the dual of the electromagnetic field tensor. The tensor $h^{\mu\nu}$ is the projection operator onto the spatial slice orthogonal to $u^\mu$, $h^{\mu\nu} = g^{\mu\nu} + u^\mu u^\nu$. The four-vector $q^\mu$ is a heat flux and the tensor $\Pi^{\mu\nu}$ models viscous transport of momentum. The model ignores bulk viscosity and resistivity. The equations governing $\rho$, $u$, and $u^\mu$ are given by the usual conservation equations,

$$\nabla_\mu N^\mu = 0$$  \hspace{1cm} (28)

$$\nabla_\mu T_{\mu\nu} = 0.$$  \hspace{1cm} (29)

Expanding the covariant derivative $\nabla_\mu$ in a coordinate basis,

$$\partial_i (\sqrt{-g} \rho u^i) + \partial_i (\sqrt{-g} P_g u^i) = 0$$  \hspace{1cm} (30)

$$\partial_i (\sqrt{-g} T_{\mu i}^\nu) + \partial_i (\sqrt{-g} T_{\nu i}^\mu) = \sqrt{-g} T_{\mu i}^\nu \Gamma^\lambda_{\nu \iota}$$  \hspace{1cm} (31)

where (30) has been obtained from (28) by scaling with $m_i$. The equations governing the components of the magnetic field
three-vector $B^i$ are given by the induction equation in the ideal MHD limit

$$\partial_t (\sqrt{-g} B^i) + \partial_j (\sqrt{-g} (b^i u^j - b^j u^i)) = 0. \quad (32)$$

The heat flux $q^i$ and the shear stress $\Pi^{\mu \nu}$ that appear in the total stress tensor (27) are

$$q^i = q \tilde{b}^i \quad (33)$$

$$\Pi^{\mu \nu} = -\Delta P \left( \tilde{b}^\mu \tilde{b}^\nu \frac{1}{3} b^{\mu \nu} \right). \quad (34)$$

where the scalar $q$ is the magnitude of the heat flux that flows parallel to the magnetic field lines and the scalar $\Delta P = P_2 - P_1$ is the pressure anisotropy, i.e., the difference in pressures perpendicular $P_2$ and parallel $P_1$ to the magnetic field. The above forms of the heat flux $q^i$ and the shear stress $\Pi^{\mu \nu}$ have been derived by assuming that the distribution functions of all species are gyrotropic, which is accurate in the limit that the Larmor radii are much smaller than the system scale. The evolution of $q$ and $\Delta P$ are given by

$$\frac{dq}{d\tau} = -\frac{q - q_0}{\tau_R} - \frac{q}{2} \frac{d}{d\tau} \log \left( \frac{\tau_R}{\chi P^2} \right) \quad (35)$$

$$\frac{d\Delta P}{d\tau} = -\frac{\Delta P - \Delta P_0}{\tau_R} - \frac{\Delta P}{2} \frac{d}{d\tau} \log \left( \frac{\tau_R}{\nu \rho P} \right) \quad (36)$$

where $d/d\tau = u^\mu \nabla_\mu$ and

$$q_0 \equiv -\rho \tilde{b}^i (\nabla_\mu \Theta + \Theta a_\mu) \quad (37)$$

$$\Delta P_0 \equiv 3 \rho \nu (\tilde{b}^i \tilde{b}^j \nabla_\mu u_{\mu i} - \frac{1}{3} \nabla_\mu u^\mu) \quad (38)$$

Here, $\Theta = P/\rho \equiv k T/m_e c^2$ is the ion temperature, $a^\mu \equiv u^\nu \nabla_\nu u^\mu = u^\nu \partial_\nu u^\mu + \Gamma^\nu_{\mu \lambda} u^\lambda u^\mu$ is the four-acceleration, and $\chi, \nu$ are the ion thermal and viscous diffusion coefficients respectively. The equations for the heat flux $q$ and the pressure anisotropy $\Delta P$ are obtained by enforcing the second law of thermodynamics. The result then is that $q$ and $\Delta P$ relax to $q_0$ and $\Delta P_0$ over the timescale $\tau_R$, with the additional term in (35) and (36) being a higher order (if $q \sim \epsilon \ll 1$, then, $q d \log(q\tau_R/(\chi P^2))/d\tau \sim \epsilon^2$ and similarly for $\Delta P$). The terms $q_0$ (37) and $\Delta P_0$ (38), which $q$ and $\Delta P$ relax to, respectively, are covariant generalizations of the Braginskii (1965) closure, which the model reduces to in the limit where the relaxation timescale $\tau_R \rightarrow 0$. The above Equations (35) and (36) can be rescaled and written as

$$\nabla_\mu (\tilde{q} u^\mu) = \frac{\tilde{q} - \tilde{q}_0}{\tau_R} + \frac{\tilde{q}}{2} \nabla_\mu u^\mu, \quad (39)$$

$$\nabla_\mu (\tilde{\Delta P} u^\mu) = -\frac{\tilde{\Delta P} - \tilde{\Delta P}_0}{\tau_R} + \frac{\tilde{\Delta P}}{2} \nabla_\mu u^\mu, \quad (40)$$

with

$$\tilde{q} = q \left( \frac{\tau_R}{\chi \rho \Theta} \right)^{1/2} \quad (41)$$

$$\tilde{\Delta P} = \Delta P \left( \frac{\tau_R}{\nu \rho \Theta} \right)^{1/2}. \quad (42)$$

These rescaled equations are crucial to our numerical implementation. Equations (35) and (36) have higher order terms $q/2 d (\log (\tau_R/(\chi P^2)))/d\tau$ and $\Delta P/2 d (\log (\tau_R/(\nu \rho P)))/d\tau$, which we find are numerically difficult to handle in low density regions. If these terms are ignored, the positivity of entropy production is no longer guaranteed. However, the rescaled Equations (39), and (40) do include these terms, and using these equations guarantees adherence to the second law of thermodynamics (up to truncation error in the numerical solution), and also leads to well behaved numerical solutions.

To conclude, the model evolves, (1) the ion rest-mass density $\rho$, (2) the total internal energy density $u$, (3) the spatial components $u^i$ of the four-velocity $u^\mu$, (4) the components of the magnetic field three-vector $B^i$, (5) the ion heat flux along magnetic field lines $q$, and (6) the ion pressure anisotropy $\Delta P$, for a total of 10 variables. The governing equations are the continuity Equation (30) for $\rho$, the energy and momentum conservation Equation (31) for $u$ and $u^i$, respectively, the induction Equation (32) for $B^i$, and the relaxation Equations (39), and (40) for $q$ and $\Delta P$ respectively. The inputs to the model are the transport coefficients $\chi$, the thermal diffusivity, and $\nu$, the kinematic viscosity. A closure scheme for $\chi$ and $\nu$ as a function of the relaxation timescale $\tau_R$ is described in Chandra et al. (2015). The scheme accounts for the presence of kinetic plasma instabilities at subgrid scales, which are prevalent in weakly collisional/collisionless plasmas. In that closure, we set $\chi = \phi \nu^2 \tau_R$ and $\nu = \psi \nu^2 \tau_R$, where $\phi$ and $\psi$ are non-dimensional numbers $\sim 1$, $c_s$ is the sound speed, and the damping timescale $\tau_R$ models the effective collision timescale for ions due to kinetic plasma instabilities.

### 4.1. Wave Speeds

The approximate Riemann solvers allowing us to capture shocks in grm require at least an upper bound on the characteristic speeds (see Section 2.4.2). The speeds control the amount of numerical dissipation introduced in the evolution. To minimize numerical dissipation, the speed estimates should be as close as possible to the true characteristic speeds, but for stability the estimates should be an upper bound on the true speeds. In ideal hydrodynamics, the characteristic speeds of the system are known analytically, but this is no longer the case for even ideal MHD. The characteristic speeds of our EMHD model can be found numerically, but this requires finding the largest and smallest zeroes of a tenth-degree polynomial on both sides of every cell face. To avoid this expensive operation, we instead follow the methods often implemented in the ideal MHD simulations, which use the HLLE or LLF Riemann solvers, and consider an upper bound on the maximum wave speed in the fluid frame, $v_{max}$. We can then obtain upper bounds on the maximum right-going and left-going wave speeds by computing the grid-frame velocity of waves propagating at $\pm v_{max}$ in the rest frame of the fluid, in the direction along which the flux is being computed. A more detailed discussion of the wave speeds of the EMHD model is provided in Chandra et al. (2015). Here we will only note that we use the practical upper bound

$$v_{max}^2 = c_s^2 + v_A^2 - c_i^2 v_A^2 \quad (43)$$

where $v_A$ is the usual Alfvén speed and $c_i$ is a correction to the sound speed including the effects of heat conduction and
viscosity:
\[
\nu_2 = \frac{b^2}{\rho + \gamma u + b^2}
\]
(44)
\[
\nu_2 = \frac{\gamma (\gamma - 1) u}{\rho + \gamma u}
\]
(45)
\[
\tilde{v}^2 = \frac{1}{2} \left(c_s^2 + v^2_q + \sqrt{c_s^4 + v^4_q + 4v^2_{dp}}\right).
\]
(46)

The corrections \(v_q\) and \(v_{dp}\) to the sound speed \(c_s\) are
\[
v^2_q = (\gamma - 1) \frac{\Delta}{\tau_R},
\]
(47)
\[
v^2_{dp} = \frac{4\nu}{3\tau_R}.
\]
(48)

With the closure scheme in Chandra et al. (2015), the speed \(\tilde{c}_s\) simplifies to
\[
\tilde{c}_s^2 = \frac{c_s^2}{2} \left(1 + (\gamma - 1)\phi + \sqrt{(1 + (\gamma - 1)^2\phi^2 + \frac{8\psi}{3}}\right).
\]
(49)

From this equation and the inequality \(c_s^2 \leq (\gamma - 1)\), we can also derive conditions on \(\psi\) and \(\phi\), which guarantee \(v^2_{\text{max}} < 1\). This is a sufficient, but not necessary, condition for the system to be causal and hyperbolic. For \(\psi = \phi\), and the standard choice of \(\gamma = 4/3\) (resp. \(5/3\)), we find \(\psi_{\text{max}} \approx 1.3\) (resp. \(\psi_{\text{max}} \approx 0.29\)). As, at the level of the Riemann solver, we do not assume a specific closure scheme in grim, we implement Equation (46) for \(\tilde{c}_s\), and not the simplified version.

4.2. Constrained Transport

A crucial ingredient for the evolution of the induction Equation (32) is the preservation of the zero monopole constraint \(\nabla \cdot \mathbf{B} = 0\). Naive evolution leads to uncontrolled growth of the constraint, resulting in numerical instabilities. Constrained transport schemes (Evans & Hawley 1988) exactly preserve a specific numerical representation of the constraint, i.e., the violations are at machine tolerance. We use a version of constrained transport by Tóth (2000), the flux-CT scheme, where the magnetic fields are co-located with the fluid variables, at the cell centers.\(^7\) They then evolve by the same routines in a finite volume sense, with the “fluxes” being the electric fields (up to a sign), which for the EMHD model (just as in ideal MHD) are \(F^l = \sqrt{-g} (b/u' - b/u^3)\). At the end of the update, the face-centered fluxes \(F^l_{\text{face}}\) (i.e., the electric fields) obtained from the Riemann solver are averaged to the edges to get the edge centered fluxes \(F^l_{\text{edge}}\). The edge centered fluxes are then averaged to get new face-centered fluxes \(\bar{F}^l_{\text{face}}\), which are then used to evolve the volume averaged magnetic fields \(\int \mathbf{B}' d\mathbf{A}'\). The simple averaging procedure we use \(F^l_{\text{face}} \rightarrow \bar{F}^l_{\text{edge}}\) is the original Tóth (2000) formulation, which is also being used in the harmc code, and lacks upwinding information (see Gardiner & Stone 2005 for a discussion on the limitations of this approach).

5. Implementation Details

We now discuss the implementation of the algorithms described in the previous sections. grim is written in C++, with a modular library architecture. Different components such as spatial reconstruction, the Riemann solver, boundary conditions, and evaluation of the metric and related quantities are all separate libraries. Each library has automated unit tests to ensure robustness against inadvertent programmer errors.\(^8\)

grim is designed to run on existing, as well as upcoming architectures. It has been tested and benchmarked on CPUs as well as on Nvidia and AMD GPUs. In (Section 7), we formulate a performance model, and describe the specifications of a machine that grim is most sensitive to. Guided by the model, we have optimized grim to achieve a significant fraction of machine peak on both CPU and GPU systems.

5.1. Dependencies

grim is built on top of the PETSc (Balay et al. 2016) library to handle distributed memory parallelism, and the ArrayFire (Yalamanchili et al. 2015) library for shared memory parallelism within a node. The C++ vector abstractions from ArrayFire allow grim to run on a variety of computer architectures (CPUs and GPUs) using the same code. We discuss this in detail in Section 5.2 and then describe how we integrate PETSc and ArrayFire to achieve architecture agnostic distributed memory parallelism in Section 5.3.

5.2. Architecture Agnostic Code

There are now several supercomputers\(^9\) that, in addition to CPUs, have accelerators such as GPUs. The programming models for these two architectures are different. We are able to write a single code that runs on both architectures by performing operations within a node using the array data structure from the ArrayFire library. Operations to be performed on an array are written down in a vector notation. For example, to add array \(A\), array \(B\), and write to array \(C\), each of which hold multidimensional data, we write \(C = A + B\) to perform the operation over the entire domain. At runtime, ArrayFire detects the available compute architectures on the node, and fires kernels customized to that architecture, using either an OpenCL, CUDA, or CPU backend.

To use a vector notation also significantly simplifies the code. The entirety of our implementation of the nonlinear solver Section 3, including the Jacobian assembly Section 3.1, the linear inversion (21), and the quadratic backtracking line search Section 3.2 is 250 lines (including comments).

All the mathematical operations that need to be performed in grim can be divided into two categories, local operations that operate point-wise, and non-local operations that require data from adjacent grid zones, such as reconstruction. We describe how both of these are implemented using the vector notation.

5.2.1. Local Operations

A majority of calculations in grim such as computing the conserved variables \(U(P)\), the fluxes \(F^{l,2,3}(P)\), and the various source terms \(S^{l,E}(P), S(\partial_t P)\) involve point-wise operations. These are easily implemented in vector notation, with certain caveats. As will be described in Section 7, the speed of vector–vector operations is set completely by the available memory bandwidth of the system, and therefore it is crucial to maximize the effective bandwidth.

---

\(^7\) We are currently testing a version that uses a face-centered formulation.

\(^8\) At present, 75 units tests.

\(^9\) www.top500.org
We illustrate what effective bandwidth means with the following computation: we have a contravariant four-vector \( u^\mu \), that we want to transform to a covariant four-vector \( u_\mu \), using 

\[
 u_\mu = g_{\mu\nu} u^\nu,
\]

where \( g_{\mu\nu} \) is the metric. Converting to computer code in vector notation, we have

```c
for (int mu = 0; mu < 4; mu++)
    uCov[mu] = 0;
for (int nu = 0; nu < 4; nu++)
    uCov[mu] += gCov[mu][nu] * uCon[nu];
```

Each of \( uCov[mu] \), \( uCov[mu] \), \( gCov[mu][nu] \), \( uCon[nu] \) is an array of size \( N_1 \times N_2 \times N_3 \), where \( N_1, N_2, \) and \( N_3 \) are the number of grid zones in the \( X^1, X^2, \) and \( X^3 \) directions, respectively, on each node. The operation \( uCov[mu] += gCov[mu][nu] * uCon[nu] \), occurs over all \( N_1 \times N_2 \times N_3 \) grid zones.

Notice that \( uCov[nu], nu = 0, 1, 2, 3 \), is being read for the computation for each of \( uCov[mu] \), \( nu = 0, 1, 2, 3 \). For grid sizes that exceed the cache, as is the case with production science runs, this involves reads from slow global memory and is therefore a performance bottleneck. In this case, the computation of \( uCov[mu] \) involves 32 global reads (16 for \( gCov[mu][nu] \), and \( 4 \times 4 = 16 \) for \( uCon[nu] \)). An optimal implementation would involve 20 global reads, with only \( 4 \) reads for \( uCon[nu] \). Therefore, the effective bandwidth achieved is only \( 20/32 \approx 0.62 \) of the ideal value.

Thus, while the abstraction of mathematical operations using vector notation allows for computation to be performed on a wide variety of computer architectures, further innovation is required to ensure optimality of the computation.

The feature that enables near-optimal performance of pointwise vector computations is ArrayFire’s lazy evaluation using its Just-In-Time (JIT) compiler. To avoid multiple reads of the memory, the operations that need to be performed on the arrays \( uCov[mu] \) are queued, instead of being immediately executed (known as eager evaluation, as is usually the case). Execution occurs using \( \text{eval}(uCov[0], uCov[1], uCov[2], uCov[3]) \). The JIT analyzes the common dependencies between all four arrays and fires a single kernel without any redundant reads and writes. In the above example, this leads to a single read for \( uCov[mu] \) instead of four separate reads. Our measurements indicate that this leads to architecture independent optimal effective bandwidth, which is crucial to the performance of our nonlinear solver. We discuss this further in Section 7.

### 5.2.2. Non-local Operations

Operations such as reconstruction and interpolation can be thought of as non-local operations because they operate on stencils of non-zero width, as opposed to point-wise local operations that operate on stencils of zero width. Non-local operations are performed using discrete convolutions. The abstraction of finite differences as discrete convolutions has two advantages: (1) it allows for an architecture agnostic code since all we require is an optimized convolution routine for CPUs and GPUs, and (2) there are indeed optimized convolution routines for both these architectures because convolutions are crucial to image processing.

A discrete convolution of input data \( g \), with a filter \( f \), at a point \( n \in [0, N) \) is defined as

\[
 (f * g)[n] = \sum_{m=-M}^{M} f[m] g[n-m]
\]

where \( g \) is an array of size \( N \), and the filter \( f \) has a stencil width \( 2M + 1 \) with an extent of \( [-M, -M + 1, \ldots, 0, \ldots, M - 1, M] \). Forward differences \( dg_+[n] = g[n+1] - g[n] \) are computed using \( f = \{1, -1, 0\} \), while backward differences \( dg_-[n] = g[n] - g[n-1] \) are computed using \( f = \{0, 1, -1\} \). Central differences \( dg = g[n+1] - g[n-1] \) are simply \( dg = dg_+ + dg_- \).

We use the optimized \texttt{convolve} function provided by ArrayFire that takes in an input array \( g \) of size \( N \), and a set of \( P \) filters \( f_1, f_2, \ldots, f_P \), and simultaneously operates all filters over the input data to return an array \( h \) with dimensions of \( N \times P \). The array \( h \) then contains the forward \( dg_+ \), and backward differences \( dg_- \) along a specified direction, over the entire domain. The combination of these two with vectorized conditional operators such as \( c = \min(a, b) \) allows us to implement the slope limiters that are required for the reconstruction operation.

### 5.3. Parallelization Infrastructure

One of the many mundane tasks involved in writing a finite volume code is the allocation of memory and initialization of several \( N_1 \times N_2 \times N_3 \times N_{\text{var}} \) arrays, where \( N_1, N_2, \) and \( N_3 \) are the number of grid zones along \( X^1, X^2, \) and \( X^3 \) directions, respectively, and \( N_{\text{var}} \) is the number of variables at each grid zone.

In addition to the memory allocation, there are several other functions the code needs for (1) partitioning the data across several nodes in a distributed memory cluster, (2) communication of ghost zones between nodes that share the same boundary, and (3) parallel file input and output that works with data spread over several nodes. To do all of the above, we created the grid class, which forms the backbone of \texttt{prim}.

#### 5.3.1. Parallelization

An instance of the grid class is created using \texttt{grid prim} \((N1, N2, N3, Ng, \text{dim}, Nvar)\)\(^{10}\), where \( N1, N2, \) and \( N3 \) are the number of grid zones along \( X^1, X^2, \) and \( X^3 \) directions, respectively, \( Ng \) is the number of ghost zones required, \( \text{dim} \) is the dimension, and \( Nvar \) is the number of variables at each zone. This builds a structured grid, performs domain decomposition using \texttt{PETSc} over a chosen number of distributed nodes, and creates \texttt{prim.vars[0], prim.vars[1], \ldots, prim.vars[Nvar-1]}, each of which is an array from the ArrayFire library, which lives on either CPUs, or GPUs, depending on the node architecture.

Each array is a contiguous block of memory of size \((N1Local + Ng) \times (N2Local + Ng) \times (N3Local + Ng)\), where \( N1Local, N2Local, \) and \( N3Local \) are the local sizes of the domain on each node. This arrangement of variables in memory is known as Strut of Arrays (SoA), leading to vectorized point-wise operations, and contiguous memory accesses. This results in optimal memory bandwidth

---

\(^{10}\) For the exact form of the definitions, please refer to the source code.
usage, which as we discuss in Section 7 determines grim’s performance.

Communication of ghost zones is performed by simply calling `prim.communicate()`, which will exchange ghost zone data of all Nvar variables in `prim` using MPI. The `communicate` function works independent of where the data lies, whether on the host CPU or attached GPU. This function works independent of where the data lies, whether on the host CPU or attached GPU.

Communication of ghost zones is performed by simply calling `prim.communicate()`, which will exchange ghost zone data of all Nvar variables in `prim` using MPI. The `communicate` function works independent of where the data lies, whether on the host CPU or attached GPU.

We have benchmarked grim on clusters with varied architectures. On the Stampede supercomputer, using NVIDIA K20 GPUs, grim evolves 138,000 grid zones/s/GPU, with 64 × 64 × 64 zones per GPU. On the CPU nodes, which have a 16 core (2 sockets × 8 cores each) Intel Xeon E5-2680 CPU, and the same resolution per node, the performance is 48,000 zones/s/node. grim scales well on both CPU and GPU machines, with ~93% weak scaling up to 4096 CPU cores on Stampede, and 256 GPUs on Bluewaters (see Figure 3).

The primary difference in speed when using grim on GPUs, as compared to CPUs is due to the higher memory bandwidth available on GPUs. The typical accessible bandwidths on CPUs are 140 GB s\(^{-1}\), while on GPUs it is 50 GB s\(^{-1}\) (using all cores on all sockets). Based on this, we expect a single GPU to be 2–3 times faster than a multicore CPU for our implementation.

### 6. Performance and Scaling

We have benchmarked grim on clusters with varied architectures. On the Stampede supercomputer, using NVIDIA K20 GPUs, grim evolves 138,000 grid zones/s/GPU, with 64 × 64 × 64 zones per GPU. On the CPU nodes, which have a 16 core (2 sockets × 8 cores each) Intel Xeon E5-2680 CPU, and the same resolution per node, the performance is 48,000 zones/s/node. grim scales well on both CPU and GPU machines, with ~93% weak scaling up to 4096 CPU cores on Stampede, and 256 GPUs on Bluewaters (see Figure 3).

The primary difference in speed when using grim on GPUs, as compared to CPUs is due to the higher memory bandwidth available on GPUs. The typical accessible bandwidths on CPUs are 140 GB s\(^{-1}\), while on GPUs it is 50 GB s\(^{-1}\) (using all cores on all sockets). Based on this, we expect a single GPU to be 2–3 times faster than a multicore CPU for our implementation.

### 7. Performance Model

The performance numbers quoted in Section 6 are experimental results. They do not give information regarding the efficiency of our implementation of the algorithms described in Sections 2 and 3. In order to do so, we require a performance model to benchmark against.

Performance of a code on a given machine is broadly set by two factors, the algorithm, and the implementation. Our root finder (Section 3) while allowing for the exploration of a broad range of fluid-like theories is much slower (a factor of ~50 slower per CPU core) than the schemes (Noble et al. 2006) used for primitive variable inversion in ideal relativistic fluids. As a result, the dominant cost in grim (~90%) is the nonlinear solver, with the sum of the reconstruction procedure, and the Riemann solver taking only ~5% of the time. Therefore, we focus our efforts on understanding the costs involved in the nonlinear solver.

The nonlinear solver involves the following three steps (1) Jacobian assembly, (2) solution of a block diagonal linear system, and (3) line search. For the linear solver, we use vendor-provided LAPACK routines, which we assume are already optimized. Therefore, we only consider the Jacobian assembly and the line search, both of which are performed by repeated calls to the residual function R(P). Given a guess for the Nvar primitive variables P, the Nvar × Nvar Jacobian J(P) (Equation (21)) is assembled using Nvar calls to the residual function R(P) (Equation (19)) that returns a vector of size Nvar. Similarly, the line search algorithm only depends on the residual function, through it norm f(λ) = ||R(P + λS)||\(^2\) (Equation (24)). Thus, it is sufficient to analyze the operations involved in the residual function.

#### 7.1. Residual Assembly

Consider the assembly of the residual R(P\(_{n+1/2}\)). It is assembled with calls to functions that compute the conserved variables U(P) and the source terms S(P), each of which return a vector of size Nvar. The other terms in the residual, U(P), \(f_1^{2,3}(P_\alpha, S^\alpha(P_\alpha), A^\alpha(P_\alpha)\partial P_\alpha\), only involve \(P_\alpha\), the primitive variables at the previous time step, and are precomputed outside the residual assembly. Therefore, the performance of the residual computation, and hence the Jacobian assembly, and the line search are set by the performance of the functions to compute U(P), S(P), and \(\partial P\). We now discuss the main factor that determines the runtime of these functions, the memory bandwidth of the system.

#### 7.1.1. Primary Architectural Bottleneck

Consider the computation of the fluid conserved variables U(P) \(\equiv T^0\), with an ideal MHD stress tensor, for brevity. The computation requires the density \(\rho \equiv \rho\), internal energy \(u \equiv u\), pressure \(P \equiv P\), the four-velocities \(u^\mu \equiv u_{\text{Con}}\), \(u_\mu \equiv u_{\text{Con}}\), the magnetic field four-vectors \(b^\mu \equiv b_{\text{Con}}\), \(b_\mu \equiv b_{\text{Con}}\), and the magnetic pressure \(b^2 \equiv b_{\text{Sqr}}\).

```c
for (int nu = 0; nu < 4; nu++)
{
    T[0][nu] = (rho + u + P + bSqr) * uCon[0] * uCon[nu] + (P + bSqr/2) * delta(0, nu)
    - bCon[0] * bCon[nu];
}
```

where \(\delta^\mu_\nu\) is the Kronecker delta.

The above code has a total of 11 floating point operations, 14 reads \(\rho, u, P, b_{\text{Sqr}}, u_{\text{Con}}, b_{\text{Con}}\), and four writes T[0][nu]. The total time taken to execute the above code is the time taken to load the data, perform the floating point operations, and finally write the data. Therefore, the total time taken is

\[
t_{\text{total}} = (N_{\text{reads}} + N_{\text{writes}} + N_{\text{reads}})N
\]

where \(N_{\text{reads}}, N_{\text{writes}}\), and \(N_{\text{reads}}\) are the total number of reads, writes, and flops performed per grid zone, \(N\) is the total number of grid zones, and \(t_{\text{read}}, t_{\text{write}},\) and \(t_{\text{flops}}\) are the times taken by the machine to perform a single read, write, and a floating point
operation respectively. The parameters \( t_{\text{read}}, t_{\text{write}}, \) and \( t_{\text{flops}} \) are architecture and machine specific. The specifications are usually given in terms of floating point operations per second flops, and memory bandwidth Bytes/s. Typical peak numbers for a current CPU are 500 Gflops, and 100 GB s\(^{-1}\). For \( N \sim 10^{9} \) (and hence ignoring latency effects), these correspond to \( Nt_{\text{flops}} \sim 0.02 \text{s}, Nt_{\text{read}} \sim 1.12 \text{s}, \) and \( Nt_{\text{write}} \sim 0.32 \text{s} \). Evidently, the ratio \( (t_{\text{reads}} + t_{\text{writes}})/t_{\text{flops}} \gg 1 \). Therefore, the runtime of the above code is almost completely set by how fast the data can be transferred between the memory system and the compute units. The actual computation time is negligible, as long as \( Nt_{\text{flops}}/(Nt_{\text{read}} + Nt_{\text{write}}) \sim 1 \), which is indeed the case for all functions involved in the Jacobian assembly and the line search.

7.1.2. Effective Bandwidth Usage

Since the performance is set by the speed of memory access, we can calculate the time it should take to compute the functions \( U(P), S'(P), \) and \( S(\partial(P)) \) by simply examining the inputs \( N_{\text{reads}} \), and the outputs \( N_{\text{writes}} \) to each function. The calculation is independent of the exact operations \( P \rightarrow \{U(P), S'(P), S(\partial(P))\} \), and is given by

\[
t(s) = \left(\frac{N_{\text{reads}} + N_{\text{writes}}}{10^9}\right) \times \frac{1}{\text{Bandwidth (GB s}^{-1})} \quad (52)
\]

where \( N_{\text{reads}}, \) and \( N_{\text{writes}} \) are the number of reads, and writes of double precision variables, each of which are 8 bytes. By measuring the runtime \( t \) of each function, the effective bandwidth being used is calculated using (52).

The measured bandwidth used in each function is now normalized with that obtained from the STREAM benchmark, given by the operation \( c = a + b \), where \( a, b, \) and \( c \) are arrays of sizes equal to the local grid sizes after domain decomposition. The STREAM benchmark has \( N_{\text{reads}} = 2(a, b), N_{\text{writes}} = 1 \) \((c)\), and is a metric of the sustained bandwidth that can be obtained on a given machine. The typical value of this benchmark on GPUs is \( \sim 140 \text{ GB s}^{-1} \), whereas on CPUs it is \( \sim 50 \text{ GB s}^{-1} \) for array sizes that exceed the cache, and when using all cores on all sockets.\(^{11}\) These numbers inform us about the potential speedup of bandwidth limited operations on GPUs, compared to CPUs.

By comparing the measured bandwidth of each function to the bandwidth obtained from the STREAM benchmark, we get the efficiency of our implementation, which we find is \( \sim 70\%–80\% \) on both GPUs and CPUs. A significantly lower (\( \leq 20\% \)) value indicates that there are either superfuous memory accesses that are not accounted for, or non-contiguous memory accesses that are not vectorized. Both of these reduce the effective memory bandwidth. The high bandwidth obtained by our implementation indicates that we have accounted for leading order performance bottlenecks in the residual evaluation, leading to a near-optimal Jacobian assembly and line search.

8. Test Suite

\texttt{grim} has been tested extensively in the linear, nonlinear, special, and general relativistic regimes. The tests below are grouped according to the physical model being solved, with subsections describing individual tests.

8.1. Extended MHD

8.1.1. Linear Modes

An important check of any numerical implementation of the EMHD model is whether it can reproduce the corresponding linear theory with an error that falls off at the expected order of spatio-temporal discretization. In order to perform this test, one needs the linear theory of the EMHD model.

The governing equations of EMHD are considerably more complicated than the governing equations of ideal MHD. In particular, the inclusion of both anisotropic pressure and conduction, which are sourced by spatio-temporal derivatives projected along the magnetic field lines, make it challenging to derive the linear theory; the derivation is prone to errors if done manually. To address this issue, we have written a general linear algebra package\(^{12}\) built on top of the SageMath (2016) computer algebra system, which takes as input the governing equations of any model and generates the characteristic matrix of the corresponding linear theory. The eigenvectors of this matrix are then used as initial conditions in \texttt{grim}, and their numerical evolutions checked against the corresponding analytic solutions.

Our linear test uses a propagating mode with wave vector \( k_1 = 2\pi, k_2 = 4\pi \) misaligned with the background magnetic field \( \mathbf{B}_0 = (0.1, 0.3, 0) \). Both \( \mathbf{k} \) and \( \mathbf{B} \) are misaligned with the numerical grid. We use the eigenvector tabulated in Table 1. Each of the variables are initialized as \( P = P_0 + A\delta P \exp(i(k_1x^1 + k_2x^2)), \) where \( P \) is the variable, \( P_0 \) is the background state, \( \delta P \) is the perturbed values, and \( A \) is the amplitude of the perturbation, which we set to \( 10^{-8} \). The exact solution is given by \( P = P_0 + A\delta P \exp(i(k_1x^1 + k_2x^2) + \omega t) \), where \( \omega = -0.5533585207638141-3.6262571286888425i \). The mode is both propagating and decaying, indicating the presence of dissipation.

The mode is evolved in a box with dimensions [0, 1] \( \times [0, 1] \), periodic boundary conditions, and resolutions \((N_1, N_2) = (32, 32), (64, 64)\). The diffusion coefficients are \( \chi = c_i^2\tau_R \) and \( \nu = c_i^2\tau_R \), with \( \tau_R = 1, c_i^2 = \gamma P/\rho + \gamma u \), \( P_i = (\gamma - 1)u \), and \( \gamma = 4/3 \). We compare the numerical and analytic solutions at \( t = 0.5 \). Figure 4 shows that the \( L_1 \) norm of the error falls off at the expected order.

8.1.2. EMHD Shock Solutions

In EMHD, viscosity can smooth a shock and connect the left and right states with a well-defined solution. The hyperbolic nature of the dissipation leads to new features in the shock structure, which have been qualitatively described in Chandra et al. (2015). Here, we solve the magnetic field aligned shock structure in the EMHD model as a boundary value problem (BVP) with the left and right states fixed to the values given by the Rankine–Hugoniot jump conditions. We then use this as a reference solution to check the EMHD shock solutions.

\(^{11}\) Balbusaur: http://bit.ly/28EGW4I.
obtained from grim, which solves the EMHD equations as an initial value problem (IVP; Figure 5).

The boundary value solutions are obtained using a global Newton root finder. We are looking for a steady-state time-independent nonlinear solution of the EMHD equations, and hence set the time derivatives $\partial_t \to 0$. Since we are interested in the continuous shock substructure, we approximate all spatial derivatives $\partial_x$ by central differences with a truncation error $O(\Delta x^3)$. Thus we have a set of coupled discrete nonlinear equations $R(P_i) = 0$, where $P_i$ is the primitive variables at $i = 0, 1, \ldots, N_x$, and $N_x$ is the chosen spatial resolution of the numerical grid. The system is iterated upon starting from a smooth initial guess using the Newton’s method combined with a numerical Jacobian assembled to machine precision. The iterations are continued until we achieve machine precision error $O(10^{-14})$.

The solution obtained from the IVP starting from a discontinuous initial condition (shown in Table 2), and the solution obtained from the BVP are connected by a translation. For a quantitative check of the error, we use the BVP solution as an initial condition into grim, and check for convergence after a fixed time. Figure 6 shows convergence between the two solutions as a function of resolution.

The EMHD theory has three free parameters, which we set to the following values: the relaxation timescale $\tau_R = 0.1$, the kinematic viscosity $\nu = \psi c_s^2 \tau_R$, and the thermal diffusivity $\chi = \phi c_s^2 \tau_R$, with the non-dimensional parameters $\psi = 3$ and $\phi = 5$. To get a continuous shock solution, we require that the characteristic speed of viscosity in the EMHD theory $v_{\text{char}} \sim (\nu/\tau_R)^{1/2} = \psi^{1/2} c_s$ be greater than the upstream velocity, here $v^1 = u^1/u^0$ in the left state. Thus, we require $v_{\text{char}} > v^1 \implies \psi > (v^1/c_s)^2$. For our chosen set of parameters, we have $v_{\text{char}} \approx 0.756 > v^1 \approx 0.707$, and hence we are able to resolve the shock structure. We find that the major contribution to the shock structure comes from the pressure anisotropy; the role of the heat conduction inside the shock is marginal. The EMHD theory has hyperbolic dissipation, which $q$ and $\Delta P$ relax to values $q_0 \propto \nabla_x T$, $\Delta P_0 \propto \nabla_x u_x$, over a timescale $\tau_R$. This leads to a structure of length $\sim v^1 \tau_R$ over which the dissipation builds up (Figure 5), and then reaches the relaxed values $q_0$, $\Delta P_0$. The theory has higher order corrections $\sim q u^i \nabla_x (\tau_R (\nabla P^2))$, $\Delta P u^i \nabla_x (\tau_R (\nabla u^i))$ that we expect to contribute in strong nonlinear regimes, and indeed we see that the shock structure differs as we turn these terms on and turn off (Figure 7). However, from Figure 7, we see that the differences are small. Still, their presence is required to enforce the second law of thermodynamics.

There is an upper limit to the strength of the shock that can be solved for using the EMHD model. Higher mach number shocks require a larger viscosity (or $\Delta P$) to smoothly connect the left and right states. However, the non-dimensional parameter $\psi$ cannot be arbitrarily large because of an upper bound on the associated characteristic speed $v_{\text{char}} \sim \psi^{1/2} c_s < c \implies \psi < (c/c_s)^2$. Beyond this critical value, the theory loses hyperbolicity, and eventually causality and stability. The root of this problem lies in the fact that ultimately, the theory is a second-order perturbation $\sim q^2$, $\Delta P^2$, about an equilibrium and, as the dissipative effects become stronger, the validity of the expansion breaks down.

What happens if we do not resolve the shock? In astrophysical applications, this is almost always the case since there is a large separation between the MHD, and the kinetic spatio-temporal scales. The pressure anisotropy $\Delta P$ is limited to the values allowed by the saturation of kinetic instabilities such as mirror and firehose. For example, $\Delta P \sim b^2$, where $b^2$ is the magnetic pressure. This viscosity may not be sufficient to resolve a shock. However, since grim is a conservative code, even when shocks are not resolved, the obtained solution asymptotes to the value given by the ideal fluid Rankine–Hugoniot jump conditions a few mean-free paths away from the shock.

8.1.3. Anisotropic Conduction Test

The EMHD model constrains heat to flow only along the magnetic field lines $q_0 \propto B^\bot (\nabla_x T + T \hat{n}_b)$. To test this, we setup a temperature perturbation in pressure equilibrium, in Minkowski spacetime with sinusoidal background magnetic field lines. The domain is a square box of size $[0, 1] \times [0, 1]$ with periodic boundary conditions. The initial conditions are

\begin{align*}
\rho & = 1 - A e^{-r^2/R^2} \\
u & = 1 \\
B^\bot & = B_0 \\
B^2 & = B_0 \sin(2\pi k x) \sqrt{2}
\end{align*}

where the amplitude of the perturbation $A = 0.2$, the radius $R = \sqrt{0.005}$, the mean magnetic field $B_0 = 10^{-4}$, and the wavenumber of the magnetic field $k = 4$. The adiabatic index is set to $\gamma = 4/3$, the relaxation timescale in the EMHD model $\tau_R = 0.1$, and the thermal diffusivity $\chi = 0.01$.

Since the initial conditions are in pressure equilibrium, they are an exact time-independent solution of the ideal MHD equations. However, the EMHD model is sensitive to temperature gradients along field lines, and hence the system

| Variable ($P_i$) | Background State ($P_0$) | Perturbed Value ($\delta P_i$) |
|------------------|---------------------------|--------------------------------|
| $\rho$           | 1                         | $-0.518225224082246 - 0.1792647678001878i$ |
| $u^i$            | 0                         | $0.5516170736393813$ |
| $u^2$            | 0                         | $0.00846322479547856 + 0.0118620226808466367i$ |
| $u^3$            | 0                         | $-0.16175466371870734 - 0.03482808082368294i$ |
| $B^1$            | 0.1                       | $-0.05973794979640743 - 0.0335170566105924i$ |
| $B^2$            | 0.3                       | $0.02986897489820372 + 0.01675837530754618i$ |
| $B^3$            | 0                         | $0.5233486841539436 + 0.04767672501939630i$ |
| $\Delta P$       | 0                         | $0.2909106062057657 + 0.02159452055336572i$ |
should evolve to a state where the plasma becomes isothermal along field lines. This outcome is shown in Figure 8, along with the transient state. As the heat flows, it excites sound waves that traverse the domain, eventually reaching the steady solution shown in the last panel in Figure 8.

8.1.4. Firehose Instability

The EMHD model, like Braginskii’s theory of weakly collisional anisotropic plasmas, is susceptible to the firehose instability. If $\Delta P < -b^2$, Alfvén waves become unstable and grow at a rate proportional to their wavenumber (see Chandra et al. 2015 for the EMHD result). To test the linear growth of a firehose-unstable mode, we consider the following initial conditions on a Minkowski background:

$$\begin{align*}
\rho &= 1 \\
u &= 2 \\
u^1 &= u^3 = 0 \\
u^2 &= A \sin(2\pi x^1) \\
B^1 &= 0.1 \\
B^2 &= B \cos(2\pi x^2)
\end{align*}$$

The Astrophysical Journal, 837:92 (21pp), 2017 March 1 Chandra, Foucart, & Gammie
Dependence of the shock substructure on the presence of higher order (HO) terms, $\sim q_0 \nabla \rho / (\gamma (\rho P))$, $\Delta P \nabla \rho / (\rho P)$, in the EMHD theory.

\[ B^3 = 0 \]  
(64)

\[ \Delta P = -0.011 \]  
(65)

with $A = 0.1628 \alpha$ and $B = 0.9867 \alpha$ chosen so that the perturbation of amplitude $\alpha$ is one of the linearly unstable Alfvén modes, with an exponential growth rate $\Gamma = 0.1036$. We artificially impose a very slow damping rate $\tau_R = 10^6$ to prevent rapid damping of the imposed pressure anisotropy $\Delta P$ toward its equilibrium value $\Delta P \approx 0$ (because the background flow has no shear).

In Figure 9, we show the evolution of the unstable mode amplitude. We observe two separate regimes of evolution. First, the unstable mode grows exponentially at the predicted rate $\Gamma$, in agreement with the linear theory. At later times, truncation error seeds perturbations on a smaller length scale, which has a much faster growth rate. Around $t = 4$, the growth of the perturbation is dominated by grid-scale modes, which grow much faster than the mode we inserted in the initial conditions, and quickly become nonlinear.

In kinetic theory, the pressure anisotropy saturates at $\Delta P \approx -b^2$. In astrophysical simulations, we similarly impose a saturation of $\Delta P$ by smoothly reducing $\tau_R$ if $\Delta P < -b^2$.

8.1.5. Hydrostatic Conducting Atmosphere

Heat conduction in curved spacetimes contains qualitatively new features when compared to Minkowski spacetime because the heat flux is driven by redshifted temperature gradients $q_0 \propto \delta_\theta (\nabla_i q + \Theta \delta_i)$. We use \( \delta_i = \Theta_i \delta_\theta \) to determine \( \Theta \) and \( \delta_\theta \), respectively. The resulting equations of hydrostatic equilibrium are

\[ \frac{\partial P}{\partial x^i} = - (\rho + u + P) \frac{\partial \ln \sqrt{-g_{00}}}{\partial x^i} \]  
(66)

\[ \frac{\partial (q \sqrt{g_{00}})}{\partial x^i} = \sqrt{-g} T^i \Gamma_{\rho \nu} \]  
(67)

\[ \frac{\partial (\Theta \sqrt{g_{00}})}{\partial x^i} = q \]  
(68)

where (66) is the momentum conservation equation in the radial direction, (68) is the energy equation, and (67) is the evolution equation for the heat flux (35), simplified in the presence of a radial magnetic field, and the absence of a radial velocity ($u' = 0$). The above equations are one-dimensional ODEs in the radial direction, which we integrate outward between two concentric spheres, starting with \((P_0, \Theta_0, q_0)\) at the inner boundary. The above equations are augmented by the ideal gas equation of state $u = P/\gamma - 1$, with $\gamma = 4/3$, and $\rho = P/\Theta$ to determine $u$ and $\rho$ respectively. The resulting (semi-)analytic solutions are then used as initial conditions in gr\_im. If the numerical implementation is correct, gr\_im should maintain the equilibrium. We consider two cases, (1) $q_0 = 0 \implies q = 0$, which is a system in thermal equilibrium, and (2) $q_0 = 0 \implies q = 0$, corresponding to a system that is conducting heat radially outward. Figure 10 shows the errors at the final time of the evolution falling off at the expected order for both cases.
8.1.6. Bondi Inflow

Spherical accretion onto a non-spinning black hole is a common test of general relativistic hydrodynamics code. It is a rare case of a non-trivial configuration for which a steady-state solution can be obtained analytically. For this test, we use as background flow the well-known solution for a spherical accretion flow around a non-spinning black hole of mass $M = 1$ due to Michel (1972). This solution has a sonic point, which we place at $r_s = 8 \, GM/c^2$. We also add a radial magnetic field $B' = 1/\sqrt{-g}$, which does not modify the hydrodynamics equilibrium.

The Bondi inflow solution has a non-trivial $u'$, the presence of a finite inflow velocity exercises all the time-independent terms in the EMHD equations for $q$ and $\Delta P$, including higher order terms that are identically zero in a hydrostatic solution. We obtain reference solutions by ignoring the backreaction of the dissipation onto the fluid flow and integrate one-dimensional ODEs in the radial direction for $q$ and $\Delta P$. We then use these solutions to check grim results obtained with backreaction turned off.

Figure 11 shows the value of the pressure anisotropy obtained at time $t = 1000 \, GM/c^3$ for a grim evolution of spherical accretion in the EMHD model, without backreaction of the pressure anisotropy onto the flow (red circles). A numerical integration of the analytical solution is shown as a solid black line. The analytical solution without higher order (HO) terms (i.e., simply damping the advected $\Delta P$ to $\Delta P_0$ on a timescale $\tau_0$) is shown as a dashed green line.
flow. In this test problem, the pressure anisotropy varies rapidly with radius, but the radial velocity of the flow is also large. For $\tau_R \gtrsim \rho / v_r$, the pressure anisotropy can thus remain significantly smaller than its Braginskii target. Figure 11 uses $\tau_R = 30$ everywhere.

8.2. Ideal MHD Tests

EMHD reduces to ideal MHD in the limit of vanishing diffusion coefficients ($\chi, \nu \to 0$), resulting in zero dissipation ($q, \Delta P \to 0$). Therefore, any code that solves the EMHD equations should also be able to handle ideal MHD. To check this, we subject grim to ideal MHD shock tests in order to check its shock capturing ability. To solve the ideal MHD equations, we simply ignore the evolution of the heat flux (39), and the pressure anisotropy (40), as well as the relevant terms in the stress-energy tensor (27). This leads to the assembly, and inversion of a $5 \times 5$ Jacobian (for the variables $\{\rho, u, u^1, u^2, u^3\}$) in the residual-based root finder, as opposed to a $7 \times 7$ Jacobian for EMHD.

We have successfully tested grim on the following ideal MHD problems (1) Komissarov (1999) shock tests, (2) relativistic Orzag–Tang (Beckwith & Stone 2011), (3) diagonal transport of an overdensity (Gammie et al. 2003), (4) low and medium magnetized cylindrical blast wave (Komissarov 1999), and (6) steady-state hydrodynamic torus (Fishbone & Moncrief 1976). The Riemann solver in grim is identical to that used in harm, therefore, we are prone to all of the known issues of the harm scheme. Specifically, the LLF flux that we use leads to

---

13 The limits $\chi, \nu \to 0$ need to be taken carefully because diffusion coefficients appear in the denominator of the higher order terms ($-q \nabla u^\alpha, \Delta P \nabla u^\alpha$) in (39), and (40), where $q \sim \sqrt{\chi}, \Delta P \sim \sqrt{\chi}$. To obtain the correct limit, rescale (39) by $\sqrt{\chi}$, and then take $\chi \to 0$, leading to $q \to 0$. The limit $\Delta P \to 0$ follows similarly.
excess diffusion at contact discontinuities when compared to schemes that explicitly model the discontinuity, like HLLC (Toro et al. 1994).

8.2.1. Komissarov Shock Tests

Komissarov (1999) formulated a series of one-dimensional nonlinear MHD solutions that are designed to check a code’s ability to correctly handle shocks and rarefactions. We ran the following cases: (1) fast shock, (2) slow shock, (3) switch-off fast, (4) switch-on slow, (5) shock tube 1, (6) shock tube 2, and (7) collision. We ran each case with 2048 grid zones in a domain \([-2, 2]\) with a minmod limiter (which in our implementation is the generalized minmod limiter, with slope set to one), and a courant factor of 0.2. As is shown in Figure 12, we correctly reproduce the expected results.

9. Applications

We describe three example applications that highlight the new physics in the EMHD model: (1) Buoyancy instabilities in weakly collisional plasmas and (2) radiatively inefficient accretion flows around supermassive black holes. We study these in global 3D domains using coordinates by Tchekhovskoy et al. (2011), A. Tchekhovskoy & R. Nemmen (2016, in preparation) that smoothly cylindrify the grid zones near the poles. This mollifies the severe time step constraints in the azimuthal (\(\phi\)) direction.

9.1. Buoyancy Instabilities

Weakly collisional plasmas are subject to instabilities not present in ideal plasmas, due to the presence of anisotropic dissipation. An ideal plasma that satisfies the Schwarzschild criterion \(ds/dz > 0\) is convectively stable. However, this is not the case when the heat flux is constrained to be parallel to magnetic field lines.

1. When the temperature decreases outward \(dT/dz < 0\) against gravity in the presence of magnetic field lines that are perpendicular to the temperature gradient,\(^{14}\) the plasma is unstable to the magneto-thermal instability (MTI; Balbus 2000).
2. When the temperature increases outwards \(dT/dz > 0\) against gravity in the presence of magnetic field lines that

\(^{14}\) When the field lines are aligned along the negative temperature gradient \(dT/dz < 0\), the system is MTI stable.
nonlinear saturation of these instabilities have been studied in-depth in the non-relativistic regime using the Braginskii (1965) model for weakly collisional plasmas. The EMHD model reduces to the Braginskii model in the non-relativistic limit, when $\tau_R \to 0$. We expect to see the MTI and the HBI in the EMHD model, and indeed we do. Below we describe the setups and the linear and the nonlinear regimes for both instabilities.

We use hydrostatic Schwarzschild-stable initial conditions in a Schwarzschild metric. We want to be able to control the sign of the temperature gradient, so that the system is either MTI or HBI unstable. To do so, we set the initial $P_0 = K\rho^\Gamma$, where $K$ is a constant, and $\Gamma$ is the polytropic index. $P$ is solved for using hydrostatic equilibrium (66), which then yields $\rho = (P_0/K)^{1/\Gamma}$ and $u$, using $u = P_0/(\gamma - 1)$. A Schwarzschild stable equilibrium requires $d\sigma/dr > 0 \implies \Gamma < \gamma$. $\Gamma$ can be changed to obtain either a positive temperature gradient $dT/dr > 0$ ($\Gamma < 1$), or a negative temperature gradient $dT/dr < 0$ ($\Gamma > 1$). For MTI, we set $K = 10^{-4}$ and $\Gamma = 4/3$, while for HBI, we set $K = 0.05$ and $\Gamma = 1/2$.

We set $\chi = c_s R$, where $R$ is radius and $c_s = \sqrt{\gamma P/\rho}$, as in Sharma et al. (2008). The EMHD model has an additional parameter, the relaxation timescale, set via $\tau_R = R/c_s$.

9.1.1. Magneto-thermal Instability

The MTI requires magnetic field lines perpendicular to the temperature gradient for maximal growth. Therefore, we perform the simulation in a half sphere $(r, \theta, \phi) \in [200, M, 300 M] \times (0, \pi) \times [0, \pi]$, and initialize with a weak azimuthal magnetic field $B_0 = 10^{-3}/\sqrt{-g}$. We use Dirichlet boundary conditions in $r$ and $\theta$ for the density $\rho$, pressure $P$, and internal energy $u$. This results in constant temperature boundaries, which continuously drive the instability. We use insulating boundary conditions for the magnetic fields, i.e., set them to zero in the boundaries. The $\phi$ boundaries are periodic for all variables.

The initial conditions have zero heat flux $q = 0$, as well as $q_0 \sim \hat{B}^\mu \nabla_\mu \Theta + \Theta a_\mu = 0$. We seed the simulation with small amplitude, $\sim 4\%$, fluctuations in $u^\mu$. These lead to small-scale corrugations of the field lines whose radial component is exponentially amplified due to the MTI (Figures 13(a) and 14(a)). Eventually, there is vigorous convection (Figure 13(a)), and a net heat flux between the radial boundaries, leading to a flattened temperature profile in the bulk of the domain (Figure 15). This is consistent with expectations from nonlinear evolution of the non-relativistic MTI.

9.1.2. Heat-flux Driven Buoyancy Instability

The HBI requires magnetic field lines to be aligned with the temperature gradient for maximal growth, and so we seed the simulation with radial field lines $B^r = 10^{-3}/\sqrt{-g}$. The spatial domain is the same 3D half sphere of the MTI setup. The boundary conditions, $\chi$, and $\tau_R$ are also identical to the MTI case.

The initial conditions have $q = 0$, but $q_0 \sim \hat{B}^\mu (\nabla_\mu \Theta + \Theta a_\mu) = 0$. The heat flux relaxes to $q_0$ over a timescale $\tau_R$, leading to a finite radial heat flux. This heat flux feeds the HBI, which grows by kinking the field lines, and leads to an exponential growth of the radial component of the magnetic field. In the saturated state, there is suppression of the

---

15 When the field lines are aligned perpendicular to the positive temperature gradient $dT/dz > 0$, the system is HBI stable.
Figure 16. Above: the intermediate state of the HBI in a global 2D setup illustrating the finite radial heat flux that develops due to the presence of radial field lines connecting the constant temperature boundaries at $R = 200$ and $R = 300$ $M$. The initial conditions have zero heat flux, and are not shown here. Below: the saturated state of the HBI that suppresses the radial heat flux of the intermediate state.
heat flux below $q_0$. Figure 16 shows the intermediate state $q_0$, which is unstable to the HBI, and the final saturated state.

9.2. Radiatively Inefficient Accretion Flow

The first astrophysical targets for the grim code are slowly accreting supermassive black holes. For a black hole with an accretion rate of $\dot{M} \lesssim 0.01M_{\text{Edd}}$ ($M_{\text{Edd}}$ = Eddington rate), we expect the surrounding accretion disk to be formed of a weakly collisional, magnetized plasma whose evolution is better approximated by our EMHD model than by the equations of ideal magnetohydrodynamics. We have already used the grim code to study the evolution of an accretion disk in the EMHD model in global, axisymmetric simulations. The current version of the grim code has also been tested on short preliminary evolutions of accretion disks in 3D, at low resolution.

In both cases, we find that the pressure anisotropy in the disk grows to values comparable to the magnetic pressure in the disk, reaching the mirror instability threshold. The closure used in our EMHD model then forces $\Delta P$ to saturate at $\approx b^2/2$. Longer, higher-resolution simulations are necessary to fully assess the impact of the EMHD model on the dynamics and energy budget of the system, and will be performed as sufficient computational resources become available.

Figure 17 shows a snapshot of such a 3D evolution at $t = 1240 \, GM/c^3$. The simulation was started from a hydrodynamical equilibrium torus Fishbone & Moncrief (1976) around a spinning black hole ($a = 0.9375$), seeded with a single loop of poloidal magnetic field. The initial amplitude of the plasma parameter $\beta = 2P/b^2$ is $\sim 100$ in the inner disk, and $\beta \gtrsim 15$ everywhere. We see growth of magnetic turbulence due to the magnetorotational instability, and growth of the pressure anisotropy to the mirror instability threshold $\Delta P = b^2/2$. The heat flux is larger than in earlier axisymmetric simulations (Foucart et al. 2016).

10. Conclusion

Low-luminosity black-hole accretion flows ($L \ll L_{\text{edd}}$) are expected to be collisionless, so anisotropic dissipative effects can be important. Understanding the disk structure, and predicting observables requires the nonlinear solutions of relativistic dissipative theories in strongly curved spacetimes. So far, numerical codes can only evolve perfect fluids, with no heat conduction or viscosity. The algorithms developed for perfect fluids do not work for relativistic dissipative theories, because dissipation in the relativistic case is sourced by spatio-temporal gradients of the thermodynamic variables, as opposed to just spatial gradients in the non-relativistic case. In this paper, we have formulated and implemented a new scheme that can handle this situation and is physics agnostic. We implement the scheme in a new code grim, which we then use to integrate the EMHD theory of anisotropic relativistic dissipation. The numerical solutions obtained have been checked against various analytic and semi-analytic solutions of the EMHD theory in both Minkowski and Schwarzschild spacetimes, in linear as well as in nonlinear regimes.

The algorithm is the same as that in Foucart et al. (2016), which has been used to study axisymmetric radiatively inefficient accretion flows, though here the code has been generalized to work in 3D and now has the ability to run on either CPUs or GPUs. Thus we are able to make full use of the various node architectures in current and future generations of supercomputers. We use a performance model to show that the implementation is near-optimal, with the code achieving a significant fraction (~70%–80%) of peak machine bandwidth. This, we show is crucial, because the performance of the nonlinear solver that is at the heart of grim is primarily dependent on the machine bandwidth.

As example applications, we have studied the magneto-thermal instability (MTI) and the heat flux driven buoyancy instability (HBI) in global 3D domains with a Schwarzschild metric, and evolved them to a nonlinear saturated state. Finally, we performed preliminary EMHD evolutions of a hydrodynamically stable torus in 3D, around a spinning (Kerr) black hole.

We thank Ben Ryan, Sasha Tchekhovskoy, Sean Ressler, Eliot Quataert, and Jim Stone for discussions as well as all the members of the horizon collaboration for their advice and encouragement (see horizon.astro.illinois.edu). The horizon collaboration is supported in part by NSF. We thank Pavan Yalamanchili at ArrayFire for his help with optimization. Support for this work was provided by NASA through an Einstein Postdoctoral Fellowship grant numbered PF4-150122 awarded to F.F. by the Chandra X-ray Center, which is operated by the Smithsonian Astrophysical Observatory for NASA under contract.
NAS8-03060. M.C. was supported by an Illinois Distinguished Fellowship from the University of Illinois and by NSF grant AST-1333612. M.C. thanks Eliot Quataert for a Visiting Scholar appointment at the University of California, Berkeley, where part of this work was done. C.F.G. was supported by NSF grant AST-1333612, a Simons Fellowship, and a visiting fellowship at All Souls College, Oxford. C.F.G. is also grateful to Oxford Astrophysics for a Visiting Professorship appointment. This work was made possible by computing time granted by UCB on the Savio cluster.

References

Balay, S., Abhyankar, S., Adams, M. F., et al. 2016, PETSc, http://www.mcs.anl.gov/petsc
Balbus, S. A. 2000, ApJ, 534, 420
Beckwith, K., & Stone, J. M. 2011, ApJS, 193, 6
Blandford, R. D., & Znajek, R. L. 1977, MNRAS, 179, 433
Braginskii, S. I. 1965, RvPP, 1, 205
Chandra, M., Gammie, C. F., Foucart, F., & Quataert, E. 2015, ApJ, 810, 162
Colella, P., & Woodward, P. R. 1984, JCoPh, 54, 174
Eckart, C. 1940, PhRv, 58, 919
Evans, C. R., & Hawley, J. F. 1988, ApJ, 332, 659
Fishbone, L. G., & Moncrief, V. 1976, ApJ, 207, 962
Foucart, F., Chandra, M., Gammie, C. F., & Quataert, E. 2016, MNRAS, 456, 1332
Gammie, C. F., McKinney, J. C., & Tóth, G. 2003, ApJ, 589, 444
Gardiner, T. A., & Stone, J. M. 2005, JCoPh, 205, 509
Harten, A., Lax, P., & van Leer, B. 1983, SIAMR, 25, 35
Hiscock, W. A., & Lindblom, L. 1983, AnPhy, 151, 466
Hiscock, W. A., & Lindblom, L. 1985, PhRvD, 31, 725
Hiscock, W. A., & Lindblom, L. 1988a, PhLA, 131, 509
Hiscock, W. A., & Lindblom, L. 1988b, in Mathematics and General Relativity: Contemporary Mathematics, Vol. 71, ed. J. A. Isenberg (Providence, RI: American Mathematical Society)
Israel, W., & Stewart, J. M. 1979, AnPhy, 118, 341
Jiang, G.-S., & Shu, C.-W. 1996, JCoPh, 126, 202
Komissarov, S. S. 1999, MNRAS, 303, 343
Liu, X.-D., Osher, S., & Chan, T. 1994, JCoPh, 115, 200
Mahadevan, R., & Quataert, E. 1997, ApJ, 490, 605
McKinney, J. C., & Gammie, C. F. 2004, ApJ, 611, 977
McKinney, J. C., Tchekhovskoy, A., & Blandford, R. D. 2012, MNRAS, 423, 3083
Michel, F. C. 1972, ApSS, 15, 153
Mosćibrodzka, M., Gammie, C. F., Dolence, J. C., Shiokawa, H., & Leung, P. K. 2009, ApJ, 706, 497
Noble, S. C., Gammie, C. F., McKinney, J. C., & Del Zanna, L. 2006, ApJ, 641, 626
Quataert, E. 2008, ApJ, 673, 758
SageMath, 2016, The Sage Mathematics Software System (Version 7.3), The Sage Developers, http://www.sagemath.org
Sharma, P., Quataert, E., & Stone, J. M. 2008, MNRAS, 389, 1815
Tchekhovskoy, A., Narayan, R., & McKinney, J. C. 2011, MNRAS, 418, L79
Toro, E. F., Spruce, M., & Speares, W. 1994, ShWav, 4, 25
Tóth, G. 2000, JCoPh, 161, 605
Yalamanchili, P., Arshad, U., Mohammed, Z., et al. 2015, ArrayFire: a general purpose GPU library, github, https://github.com/arrayfire/arrayfire
Yuan, F., & Narayan, R. 2014, ARA&A, 52, 529