




GAMERA: A Three-dimensional Finite-volume MHD Solver for Non-orthogonal Curvilinear Geometries

Binzheng Zhang¹, Kareem A. Sorathia², John G. Lyon³, Viacheslav G. Merkin² , Jeffrey S. Garretson², and Michael Wiltberger⁴

¹Department of Earth Sciences, the University of Hong Kong, Hong Kong S.A.R., People's Republic of China; binzh@hku.hk

²Applied Physics Laboratory, Johns Hopkins University, MD, USA

³Department of Physics and Astronomy, Dartmouth College, NH, USA

⁴National Center for Atmospheric Research, CO, USA

Received 2018 November 30; revised 2019 June 10; accepted 2019 July 8; published 2019 September 19

Abstract

Efficient simulation of plasmas in various contexts often involves the use of meshes that conform to the intrinsic geometry of the system under consideration. We present here a description of a new magnetohydrodynamic code, GAMERA (Grid Agnostic MHD for Extended Research Applications), designed to combine geometric flexibility with high-order spatial reconstruction and constrained transport to maintain the divergence-free magnetic field. GAMERA carries on the legacy of its predecessor, the LFM (Lyon–Fedder–Mobarry), a research code whose use in space physics has spanned three decades. At the time of its initial development, the LFM code had a number of novel features: eighth-order centered spatial differencing, the Partial Donor Cell Method limiter for shock capturing, a non-orthogonal staggered mesh with constrained transport, and conservative averaging-reconstruction for axis singularities. The capability to handle multiple ion species was also added later. GAMERA preserves the core numerical philosophy of LFM while also incorporating numerous algorithmic and computational improvements. The upgrades in the numerical schemes include accurate grid metric calculations using high-order Gaussian quadrature techniques, high-order upwind reconstruction, non-clipping options for interface values, and improved treatment of axis singularities. The improvements in the code implementation include the use of data structures and memory access patterns conducive to aligned vector operations and the implementation of hybrid parallelism, using MPI and OMP. GAMERA is designed to be a portable and easy-to-use code that implements multidimensional MHD simulations in arbitrary non-orthogonal curvilinear geometries on modern supercomputer architectures.

Key words: magnetohydrodynamics – methods: numerical – plasmas

Supporting material: animations

1. Introduction

Numerical magnetohydrodynamics (MHD), which solves the equations of ideal or nonideal MHD, is used extensively in research areas such as basic plasma physics, astrophysics, solar physics, geospace environment modeling, planetary sciences, and space weather applications. Therefore, the development of accurate, multidimensional, general-purposed MHD solvers along with comprehensive descriptions of their implementation is important for building powerful computational tools and also for the advancement of scientific understanding and education of the new generation of scientists. In the past two decades, many general-purposed MHD codes have been developed by research teams from various scientific research communities, including but not limited to, the ZEUS code (Stone & Norman 1992), the Athena code (Stone et al. 2008), the BATS-R-US code (Powell et al. 1999), the VAC code (Tóth 1997), the Nirvana (Ziegler 2008), the RAMSES (Fromang et al. 2006), the PLUTO code (Mignone et al. 2007), the AstroBEAR (Cunningham et al. 2011), the FLASH code (Dubey et al. 2008), and the PENCIL code (Dobler & Stix 2006). One of the MHD codes that has been widely used within the space physics community is the Lyon–Fedder–Mobarry (LFM) code (Lyon et al. 2004).

The LFM MHD code, initially developed at the Naval Research Laboratory in the early 1980s, is one of the pioneers of solving three-dimensional MHD equations in non-orthogonal curvilinear geometry with high-quality advection schemes (Lyon et al. 1981, 2004; Brecht et al. 1982). The MHD Kernel of the

LFM code had a number of novel features: an eighth-order centered spatial reconstruction method to reduce numerical diffusion, a universal-type nonlinear limiter to capture shocks (Hain 1987), a non-orthogonal spherical mesh with a high-order constrained transport algorithm to maintain zero magnetic divergence at round-off, a fix for axis singularities, and multi-fluid extensions to handle multiple ion species (Brambles et al. 2011). The LFM code is well known as a global terrestrial magnetosphere model, which has been one of the workhorses for geospace research and space weather applications (Luhmann et al. 2004). In the past decade, as the backbone of a whole geospace model, the LFM has been coupled to other first-principles codes, e.g., the magnetosphere-ionosphere coupler/solver code for high-latitude ionospheric electrodynamics (Merkin & Lyon 2010), the Rice Convection Model of the inner magnetosphere (Pembroke et al. 2012), the NCAR thermosphere-ionosphere model for upper atmospheric dynamics (Wiltberger et al. 2004), and the ionosphere polar wind model (Varney et al. 2016) for collisionless heavy ion transport. Beyond geospace research, the LFM code has also been adapted to be an inner heliosphere model (Merkin et al. 2011), a global heliosphere model (McNutt et al. 1999), an unmagnetized planetary magnetosphere model for Venus (Kallio et al. 1998), a giant planetary magnetosphere model for Jupiter (Zhang et al. 2018), and basic plasma simulations for magnetic reconnection (Merkin et al. 2015). The code is actively used by research teams in the space physics community, and the list of applications developed based on the LFM code is still growing.

For a code to maintain its place in computational science for almost 40 yr is a testament to the quality of its core numerics, but the age of the LFM code makes it difficult to adapt it to modern and future architectures. To address these issues, we completed a rewrite of the LFM code from scratch, re-envisioning it as *GAMERA—Grid Agnostic MHD for Extended Research Applications*. The goal of the GAMERA project was to rebuild the LFM code with two underlying design principles: retain and improve the high-heritage MHD numerics, and prepare the code for the exascale era. These improvements include (1) flexible grid specifications, (2) high-order grid metric calculations based on Gaussian quadrature, (3) a seventh-order upwind spatial reconstruction, (4) non-clipping options in the limiters, and (5) a higher-order averaging-reconstruction method for axis singularity. While based on the best elements of the LFM legacy, GAMERA is designed to be easily applicable to two- or three-dimensional MHD simulations using general curvilinear computational grids adapted to specific problems. It is highly efficient; for example, usable Earth or planetary magnetospheric simulations can be run on a laptop close to real time.

The goal of this paper is to provide a general description of the GAMERA code, including both the numerical schemes and the implementation details. The intent is that this paper will serve as a reference for future users and developers to use, modify, and eventually extend the code to their own research applications. The description contains enough details so that the numerical techniques and equations used in the MHD kernel of the GAMERA code can be easily identified, or a functionally equivalent code can be built based on the details described in this paper. An example MATLAB code with the GAMERA algorithms described in this paper is provided on Zenodo (Zhang et al. 2019a) under a Creative Commons Attribution license as a supplement for potential users to further understand the numerical algorithms. With a set of standard test problems as references, the quality of the numerical schemes used in GAMERA can be evaluated and improved when necessary. The paper is organized as follows: Section 2 describes the default MHD equations solved in the GAMERA code. Section 3 describes the detailed numerical schemes used in the GAMERA code, including grid discretization, time stepping, spatial reconstruction, flux functions, constrained transport algorithms in non-orthogonal geometry, and details about the code implementation including vectorization and parallelization. Section 4 shows results from five representative test simulation problems to demonstrate the quality of the numeric algorithms, and a summary and conclusions are included in Section 5.

2. The Basic MHD Equations

The default equation set solved in the GAMERA code is the single-fluid, normalized ideal MHD in a semi-conservative form with the plasma energy equation:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{u}) \quad (1)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla \cdot (\rho \mathbf{u} \mathbf{u} + \bar{\mathbf{I}} P) - \nabla \cdot \left(\bar{\mathbf{I}} \frac{B^2}{2} - \mathbf{B} \mathbf{B} \right) \quad (2)$$

$$\frac{\partial E_p}{\partial t} = -\nabla \cdot [\mathbf{u}(E_p + P)] - \mathbf{u} \cdot \nabla \cdot \left(\frac{B^2}{2} \bar{\mathbf{I}} - \mathbf{B} \mathbf{B} \right) \quad (3)$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \quad (4)$$

where ρ is the plasma mass density, \mathbf{u} is the plasma bulk velocity, $\bar{\mathbf{I}}$ is the unit tensor, P is the plasma thermal pressure, \mathbf{B} is the magnetic field, and $\mathbf{E} = -\mathbf{u} \times \mathbf{B}$ is the electric field based on the ideal Ohm's law. E_p is the plasma energy defined as the sum of kinetic and thermal energy

$$E_p = \frac{1}{2} \rho u^2 + \frac{P}{\gamma - 1}, \quad (5)$$

with $\gamma = \frac{5}{3}$ defined as the ratio of specific heat. The normalization of Equations (1)–(4) can be found in Appendix A.

The fluid part of the semi-conservative form of the ideal MHD equation set can be written in a compact vector form

$$\frac{\partial \mathbf{U}^C}{\partial t} = \nabla \cdot \mathbf{F}(\mathbf{U}^C) + \mathbf{M}(\mathbf{U}^C), \quad (6)$$

where \mathbf{U}^C is the vector form of the conserved density variables defined as

$$\mathbf{U}^C = \begin{pmatrix} \rho \\ \rho \mathbf{u} \\ E_p \end{pmatrix}. \quad (7)$$

$\mathbf{F}(\mathbf{U}^C)$ is the vector form of the fluid part of the ideal MHD equations (ideal hydrodynamics)

$$\mathbf{F}(\mathbf{U}^C) = - \begin{bmatrix} \rho \mathbf{u} \\ \rho \mathbf{u} \mathbf{u} + \bar{\mathbf{I}} P \\ \mathbf{u}(E_p + P) \end{bmatrix}, \quad (8)$$

and $\mathbf{M}(\mathbf{U}^C)$ is the vector form of the magnetic terms from the Lorentz force defined as

$$\mathbf{M}(\mathbf{U}^C) = - \begin{bmatrix} 0 \\ \nabla \cdot \left(\bar{\mathbf{I}} \frac{B^2}{2} - \mathbf{B} \mathbf{B} \right) \\ \mathbf{u} \cdot \nabla \cdot \left(\bar{\mathbf{I}} \frac{B^2}{2} - \mathbf{B} \mathbf{B} \right) \end{bmatrix}. \quad (9)$$

The corresponding state vector of the primitive variables \mathbf{U}^P is defined as

$$\mathbf{U}^P = \begin{pmatrix} \rho \\ \mathbf{u} \\ P \end{pmatrix}, \quad (10)$$

which is mainly used in the reconstruction module. Another important set of variables used in the MHD solver is the volume-integrated conserved densities, with a vector form defined as

$$\mathbf{U}^V = \int \mathbf{U}^C dV = \int \begin{pmatrix} \rho \\ \rho \mathbf{u} \\ E_p \end{pmatrix} dV, \quad (11)$$

which are the quantities that are actually conserved when the MHD equations are evolved.

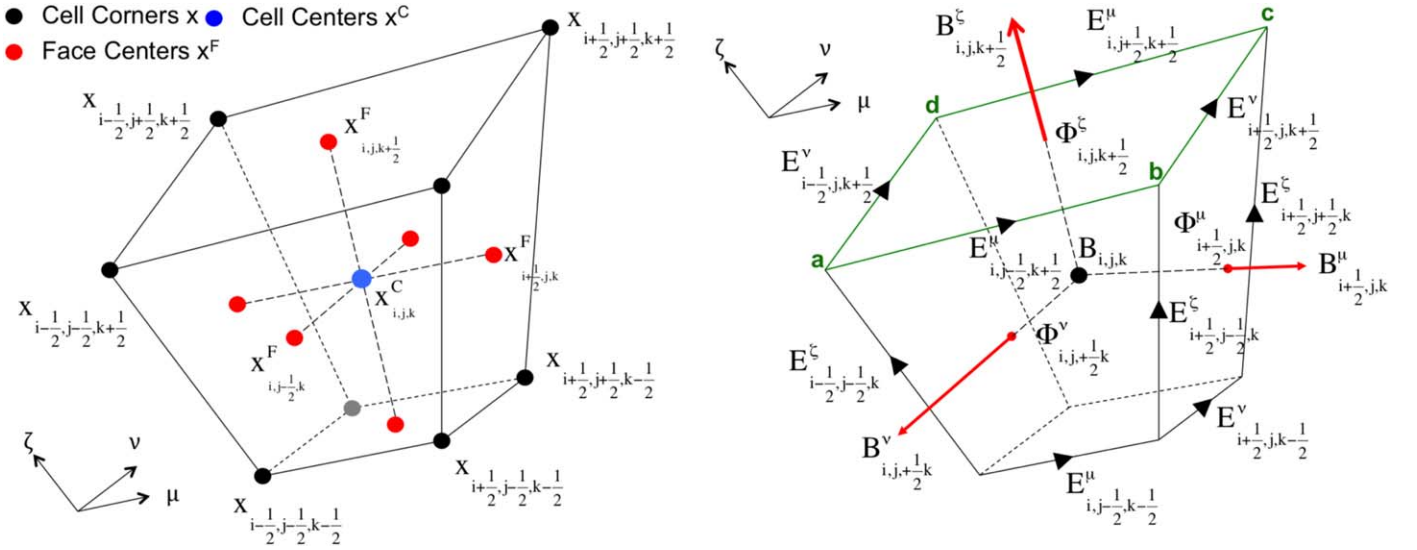


Figure 1. (Left panel) The grid definition for cell center x^C , cell face x^F , and cell corner x locations in a computational cell. (Right panel) The staggered grid definition for magnetic fluxes/fields and electric fields.

The use of the plasma energy equation has significant advantages in a number of problems, especially in modeling planetary magnetospheres with strong background magnetic fields and cold ambient plasmas (e.g., $\beta < 10^{-6}$). However, in using the plasma energy equation instead of the total energy equation, the system of equations is no longer in a fully conservative form. Nonconservative forms of the energy equation can lead to nonphysical results if nonideal processes occur, whether physical or numerical. However, Lyon et al. (2004) showed that the use of plasma energy equation does not change the Rankine–Hugoniot relations to within the numerical truncation error. The result is independent of whether or not the electric field is carried by dissipative processes through the shock. The truncation level of the numerical error is not quite as low as the round-off error from the total energy formulation, but it is sufficient to ensure that the behavior of shocks is correct to a good approximation.

3. Numerical Schemes

3.1. Finite-volume Spatial Discretization

In this section, we describe the definitions of the curvilinear computational grids, the plasma and magnetic variables, and basic metric calculations such as face area, face-normal unit vector, and cell volume used in the non-orthogonal, finite-volume MHD solver.

3.1.1. Grid Definitions

The basic cell structure defined in the finite-volume solver is that of a general hexahedron without the requirement of coordinate orthogonality as shown in Figure 1. Although the physical cells used in the solver can be general hexahedra, the grid structure is logically Cartesian in the computational space. In other words, the grid used in the MHD solver is curvilinear. In the finite-volume method, the differential form of the MHD Equations (1)–(3) are volume-integrated over individual cells. Thus, the quantities then referred to are not the point values, as they would be in a finite difference formulation, but the volume-averaged values within a computational cell. Using

Gauss’s law, the conservative part of the volume-integrated form of Equation (6) becomes a surface integral

$$\begin{aligned} \frac{\partial}{\partial t} \int_V U^c dV &= \frac{\partial}{\partial t} U^V = \int_V \nabla \cdot \mathbf{F}(U^c) dV \\ &= \oint \mathbf{F}(U^c) \cdot d\mathbf{S}, \end{aligned} \quad (12)$$

where U^V is the vector form of the volume-integrated conserved variables as defined in (11). The Lorentz terms $\mathbf{M}(U^c)$ in (6) are treated in a similar way, using an operator splitting technique, which is discussed in Section 3.2.2. The differential form of the Maxwell’s Equation (4) is integrated over cell interfaces such that the actual magnetic variable evolved in the code is the magnetic flux through cell interfaces, which is discussed in detail in Section 3.5. While it is possible to solve for the non-Cartesian representations of the velocity and magnetic field using a finite-volume technique, it generally leads to geometry-related source terms in the integral form of the MHD equations in which cell integrals must be done explicitly in (12). Therefore, in the GAMERA code, all of the cell-centered vector components are represented as *Cartesian* fields in order to avoid such source terms originating from general curvilinear coordinates. In the GAMERA code, the Cartesian components of the cell-centered velocity \mathbf{u} (u_x, u_y, u_z) and magnetic field vectors \mathbf{B} (B_x, B_y, B_z) are defined in the “Base Cartesian Coordinate System.”

In the following sections, the Base Cartesian coordinates are denoted as (x, y, z) , while the curvilinear coordinates are denoted as (μ, ν, ζ) as shown in Figure 1. In the computational space, the cell-centered index of the curvilinear grid is given by integers i, j , and k for the μ -, ν -, and ζ -directions, respectively. Thus, the corresponding cell corner and face index are represented using half integers such as $i \pm \frac{1}{2}, j \pm \frac{1}{2}, k \pm \frac{1}{2}$, respectively. Following these index notations, the spatial locations of the “cell centers” are denoted as $x_{i,j,k}^C$, and the “cell corners” are then located at $x_{i \pm \frac{1}{2}, j \pm \frac{1}{2}, k \pm \frac{1}{2}}$ given that the $\pm \frac{1}{2}$ index indicates displacement halfway between cell centers. The spatial location of the face centers at the μ -, ν -, and ζ -face are

Table 1
The Grid Definitions and Index Notations Used in the MHD Solver

Grid Variable	Grid Location	Grid Index	Notation Used in Paper
\mathbf{x} (primary)	Cell corners	$i + \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2}$	$\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$
\mathbf{x}^C	Cell centers	i, j, k	$\mathbf{x}_{i,j,k}^C$
	Face centers (μ)	$i + \frac{1}{2}, j, k$	$\mathbf{x}_{i+\frac{1}{2},j,k}^\mu$
\mathbf{x}^F	Face centers (ν)	$i, j + \frac{1}{2}, k$	$\mathbf{x}_{i,j+\frac{1}{2},k}^\nu$
	Face centers (ζ)	$i, j, k + \frac{1}{2}$	$\mathbf{x}_{i,j,k+\frac{1}{2}}^\zeta$
Volume	Cell volume	i, j, k	$V_{i,j,k}$
	Face area (μ)	$i + \frac{1}{2}, j, k$	$A_{i+\frac{1}{2},j,k}^\mu$
A^F	Face area (ν)	$i, j + \frac{1}{2}, k$	$A_{i,j+\frac{1}{2},k}^\nu$
	Face area (ζ)	$i, j, k + \frac{1}{2}$	$A_{i,j,k+\frac{1}{2}}^\zeta$
\mathbf{n}_F	Face-normal unit vector (μ)	$i + \frac{1}{2}, j, k$	$\mathbf{n}_{\mu,i+\frac{1}{2},j,k}$
	Face-normal unit vector (ν)	$i, j + \frac{1}{2}, k$	$\mathbf{n}_{\nu,i,j+\frac{1}{2},k}$
	Face-normal unit vector (ζ)	$i, j, k + \frac{1}{2}$	$\mathbf{n}_{\zeta,i,j,k+\frac{1}{2}}$

denoted as $\mathbf{x}_{i\pm\frac{1}{2},j,k}^\mu$, $\mathbf{x}_{i,j\pm\frac{1}{2},k}^\nu$, and $\mathbf{x}_{i,j,k\pm\frac{1}{2}}^\zeta$, respectively. Note that in the GAMERA code, the *primary* grid is the cell corner coordinates $\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$, while cell centers, face centers, and other grid metric calculations are derived from the cell corners. These grid metric calculations include the cell volume, face area, and face-normal unit vector. Using the indices defined for the curvilinear grid, the cell volume is denoted as $V_{i,j,k}$, the face areas at the μ -, ν -, and ζ -interface are denoted as $A_{i\pm\frac{1}{2},j,k}^\mu$, $A_{i,j\pm\frac{1}{2},k}^\nu$, and $A_{i,j,k\pm\frac{1}{2}}^\zeta$, respectively. Another set of grid metric quantity used in the MHD solver for the calculations of interface fluxes is the unit vectors of the interface normal directions, which is denoted as $\mathbf{n}_{i,j+\frac{1}{2},k}^\mu$, $\mathbf{n}_{i,j+\frac{1}{2},k}^\nu$, and $\mathbf{n}_{i,j+\frac{1}{2},k}^\zeta$ for the μ -, ν -, and ζ -interface, respectively. The grid definition and notations used in the following sections are listed in Table 1. The calculations of cell volume (V), face center locations (\mathbf{x}^F), and face area (A) use the 12th-order Gaussian quadrature and are described in Appendix B. The face-normal vector and the associated coordinate transforms are used in computing numerical fluxes through the cell interfaces, which is introduced in Section 3.4.

Based on the finite-volume discretization shown in Figure 1, for a single control volume indexed as (i, j, k) , the volume-integrated form of Equation (12) becomes

$$\begin{aligned}
 \frac{\partial}{\partial t} \mathbf{U}_{i,j,k}^V = & (A_{i-\frac{1}{2},j,k}^\mu \mathbf{n}_{i-\frac{1}{2},j,k}^\mu \cdot \mathbf{F}_{i-\frac{1}{2},j,k}^\mu \\
 & + A_{i+\frac{1}{2},j,k}^\mu \mathbf{n}_{i+\frac{1}{2},j,k}^\mu \cdot \mathbf{F}_{i+\frac{1}{2},j,k}^\mu) \\
 & + (A_{i,j-\frac{1}{2},k}^\nu \mathbf{n}_{i,j-\frac{1}{2},k}^\nu \cdot \mathbf{F}_{i,j-\frac{1}{2},k}^\nu \\
 & + A_{i,j+\frac{1}{2},k}^\nu \mathbf{n}_{i,j+\frac{1}{2},k}^\nu \cdot \mathbf{F}_{i,j+\frac{1}{2},k}^\nu) \\
 & + (A_{i,j,k-\frac{1}{2}}^\zeta \mathbf{n}_{i,j,k-\frac{1}{2}}^\zeta \cdot \mathbf{F}_{i,j,k-\frac{1}{2}}^\zeta \\
 & + A_{i,j,k+\frac{1}{2}}^\zeta \mathbf{n}_{i,j,k+\frac{1}{2}}^\zeta \cdot \mathbf{F}_{i,j,k+\frac{1}{2}}^\zeta)
 \end{aligned} \quad (13)$$

where $\mathbf{U}_{i,j,k}^V$ is the volume-integrated conservative quantities and \mathbf{F} is the numerical flux through the corresponding cell interfaces.

3.1.2. Variable Definitions

The MHD variables used in the GAMERA code are listed in Table 2. The detailed definitions and calculations for these variables are described in the following sections.

3.1.3. Plasma variables

The fluid state vector $\mathbf{U}_{i,j,k}^V$ of the mass, momentum, and energy in the GAMERA code is defined as the volume integral of the conserved densities $\mathbf{U}_{i,j,k}^C$ over the controlled volume (i, j, k)

$$\begin{aligned}
 \mathbf{U}_{i,j,k}^V &= \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} d\zeta \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} d\nu \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} d\mu \mathbf{U}^C(x, y, z) d\mu \\
 &= V_{i,j,k} \cdot \mathbf{U}_{i,j,k}^C
 \end{aligned} \quad (14)$$

Here, $\mathbf{U}_{i,j,k}^C$ are the mean densities (mass density, momentum density, and energy density) within the cell (i, j, k) . As a result, what are actually conserved in the MHD solver are the volume-integrated quantities $\mathbf{U}_{i,j,k}^V$. In the reconstruction algorithm introduced in Section 3.3.1, the high-order reconstruction method for the interface value used in the GAMERA code is based on the volume-integrated conserved variables $\mathbf{U}_{i,j,k}^V$ such that the variation of grid geometry is taken into account in the reconstruction process for nonuniform grids. The limiting step for the left and right state at an interface is applied to the primitive state vector $\mathbf{U}_{i,j,k}^P$ without the impact of variation from grid geometry. The details are described in Section 3.3.1.

3.1.4. Magnetic Fluxes and Fields

The MHD solver adapts the Yee-Grid (Yee 1966) to non-orthogonal geometries in order to enforce the conservation of the volume-integrated magnetic divergence to the round-off error, which is illustrated in the right panel of Figure 1. Therefore, the primary electric and magnetic variables are defined on a staggered, non-orthogonal grid that are not colocated with the plasma variables at the cell centers. In the GAMERA code, the primary magnetic field variables evolved by Faraday's law are the magnetic flux Φ through the cell faces. Using the Φ^μ component shown in the right panel of Figure 1 as an example, the magnetic flux $\Phi_{i+\frac{1}{2},j,k}^\mu$ threading a face in the μ -direction indexed as $(i + \frac{1}{2}, j, k)$ is calculated as

$$\begin{aligned}
 \Phi_{i+\frac{1}{2},j,k}^\mu &= \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} d\zeta \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} d\nu \mathbf{B} \cdot \mathbf{A}^\mu \\
 &= \mathbf{B}_{i+\frac{1}{2},j,k}^\mu \cdot \mathbf{n}_{i+\frac{1}{2},j,k}^\mu A_{i+\frac{1}{2},j,k}^\mu
 \end{aligned} \quad (15)$$

where $\mathbf{B}_{i+\frac{1}{2},j,k}^\mu$ is the mean magnetic field vector at the center of a cell face in the μ -direction. In the GAMERA code, since we track the magnetic flux Φ , the face-centered magnetic field vectors at the face centers are nowhere computed directly. Using constrained transport based on the staggered grid, the magnetic fluxes $\Phi^{\mu,\nu,\zeta}$ are evolved using the cell-edge-centered electric fields (or electric potential) surrounding the face flux

Table 2
Variable Definitions in the MHD Solver

Variable Type	Variable Notation	Location Defined	Property	Coordinates
Plasma State (U)	ρ	cell center	scalar	...
	\mathbf{u} (u_x, u_y, u_z)	cell center	vector	Base-(x, y, z)
	P	cell center	scalar	...
Magnetic Flux (Φ)	$\Phi^{\mu,\nu,\zeta}$ (Φ_x, Φ_y, Φ_z)	face	scalar	...
Magnetic Field (\mathbf{B})	$\mathbf{B}^{x,y,z}$ (B_x, B_y, B_z)	cell center	vector	Base-(x, y, z)
Electric Field (\mathbf{E})	$\mathbf{E}^{\mu,\nu,\zeta}$ (E_μ, E_ν, E_ζ)	edge	vector	edge-aligned-(μ, ν, ζ)

through the integral form of the Faraday's law. For example, using the $\Phi_{i,j,k+\frac{1}{2}}^\zeta$ component at the $(i, j, k + \frac{1}{2})$ cell face as an example shown in the right panel of Figure 1, according to Stokes' theorem, the integral form of Faraday's law (4) is expressed by integrating the electric field along the green contour surrounding $\Phi_{i,j,k+\frac{1}{2}}^\zeta$:

$$\frac{\partial}{\partial t} \Phi_{i,j,k+\frac{1}{2}}^\zeta = \oint_{a-b-c-d-a} \mathbf{E} \cdot d\mathbf{l}, \quad (16)$$

where \mathbf{E} is the electric field component along the edge and $d\mathbf{l}$ is the corresponding length of the cell-edge vector. The integral path on the rhs of (16) is defined as the closed contour $a-b-c-d-a$, as illustrated in the right panel of Figure 1. In the MHD solver, the contour integral is done explicitly in a piecewise way, i.e., the quantities $\mathbf{E} \cdot d\mathbf{l}$ are computed at each cell edge with the electric field \mathbf{E} calculated using high-order reconstruction schemes adapted to the cell edges.

Using the staggered discretization of the magnetic flux and electric field, it is straightforward to show that the divergence of the magnetic field $\nabla \cdot \mathbf{B}$ is conserved in an integral sense (Evans & Hawley 1988), as long as $\nabla \cdot \mathbf{B} = 0$ is satisfied initially. In other words, the action of an electric field along any cell edge leaves $\nabla \cdot \mathbf{B}$ unchanged during computations. This staggered discretization of the magnetic field allows nonlinear limiters that modify the electric field locally to still conserve the solenoidal behavior of the magnetic field, which is essential to the quality of the numerical solutions to the MHD equations.

In order to compute the Lorentz force terms $-\nabla \cdot (\bar{\mathbf{I}} \frac{B^2}{2} - \mathbf{B}\mathbf{B})$ on the bulk plasma using the magnetic flux functions described in Section 3.4, the Cartesian magnetic field vector denoted as $\mathbf{B}_{i,j,k}^{x,y,z}$ (defined in the base Cartesian coordinates) colocated at cell centers $\mathbf{x}_{i,j,k}^C$ with plasma variables are needed. To compute the cell-centered magnetic field vectors $\mathbf{B}_{i,j,k}^{x,y,z}$ using the face-centered magnetic fluxes $\Phi^{\mu,\nu,\zeta}$, consider the volume integral of $\nabla \cdot (\mathbf{r}\mathbf{B})$ within a controlled volume V given that $\nabla \cdot \mathbf{B} \equiv 0$:

$$\begin{aligned} \int_V \nabla \cdot (\mathbf{r}\mathbf{B}) dV &= \int_V (\mathbf{B} \cdot \nabla) \mathbf{r} dV + \int_V (\nabla \cdot \mathbf{B}) \mathbf{r} dV \\ &= \int_V \mathbf{B} \cdot d\mathbf{V} \\ &= \bar{\mathbf{B}} V, \end{aligned} \quad (17)$$

where $\bar{\mathbf{B}}$ is the mean magnetic field vector within a finite-volume cell, and V is the corresponding cell volume. Applying Gauss's law to the LHS of Equation (17), the volume integral is

then converted to the following face integral:

$$\int_V \nabla \cdot (\mathbf{r}\mathbf{B}) dV = \oint_S (\mathbf{r}\mathbf{B}) \cdot d\mathbf{S} = \oint_S \mathbf{r} (\mathbf{B} \cdot d\mathbf{S}). \quad (18)$$

Using Equations (17) and (18) with $\nabla \cdot \mathbf{B} \equiv 0$, the cell-centered magnetic field \mathbf{B} is computed as

$$\bar{\mathbf{B}} = \frac{1}{V} \oint_S \mathbf{r} (\mathbf{B} \cdot d\mathbf{S}) = \frac{1}{V} \sum_{S=\mu,\nu,\zeta} \mathbf{r}^S \Phi^S. \quad (19)$$

The rhs of Equation (19) is a face integral using the primary magnetic flux variables $\Phi^{\mu,\nu,\zeta}$. In the GAMERA code, a second-order approximation based on the face-centered magnetic flux is used to approximate the face integral on the rhs of Equation (19), i.e., the cell-centered magnetic fields $\mathbf{B}_{i,j,k}^{xyz}$ using estimations of magnetic flux at cell centers are computed as

$$\begin{aligned} \mathbf{B}_{i,j,k}^{xyz} &= \frac{1}{V_{i,j,k}} [(\Phi_{i+\frac{1}{2},j,k}^\mu \mathbf{x}_{i+\frac{1}{2},j,k}^\mu - \Phi_{i-\frac{1}{2},j,k}^\mu \mathbf{x}_{i-\frac{1}{2},j,k}^\mu) \\ &\quad + (\Phi_{i,j,k+\frac{1}{2}}^\nu \mathbf{x}_{i,j,k+\frac{1}{2}}^\nu - \Phi_{i,j,k-\frac{1}{2}}^\nu \mathbf{x}_{i,j,k-\frac{1}{2}}^\nu) \\ &\quad + (\Phi_{i,j,k+\frac{1}{2}}^\zeta \mathbf{x}_{i,j,k+\frac{1}{2}}^\zeta - \Phi_{i,j,k-\frac{1}{2}}^\zeta \mathbf{x}_{i,j,k-\frac{1}{2}}^\zeta)]. \end{aligned} \quad (20)$$

The above formulation is simple and works for any hexahedral cell defined by its corners. Note that $\nabla \cdot \mathbf{B} = 0$ is required in the above equation, calculating the Cartesian components of the magnetic fields located at the cell centers.

3.2. Time Stepping for Temporal Evolution

This section provides a general description of the numerical algorithms used in the GAMERA code for evolving the MHD Equations (1)–(4), including the time-stepping methods, the Courant condition calculations, the operator splitting technique for handling the magnetic terms $\mathbf{M}(\mathbf{U})$ in (6), and the framework for handling the system of equations in non-orthogonal, curvilinear geometries. Figure 2 illustrates the main algorithms of the MHD solver including the predictor/corrector steps, the fluid solver for evolving the plasma variables in Equations (1)–(3), and the Maxwell solver updating the magnetic flux/fields in Equation (4).

For time stepping, we use the Adams–Bashforth second-order scheme as the default algorithm. The predictor step is a linear extrapolation to the half-time level ($t_{n+\frac{1}{2}}$) based on the state variables of the current (t_n) and previous (t_{n-1}) time steps. For the fluid part of the MHD equations, the corrector step is a full flux calculation for the integral form of Equations (1)–(3) based on the predicted state using the Partial Interface Method (PIM) developed by Lyon et al. (2004), which is denoted as the “Fluid Solver” and introduced in Section 3.2.2. For evolving the integral form of Faraday's law, the corrector step is the calculation of electric fields

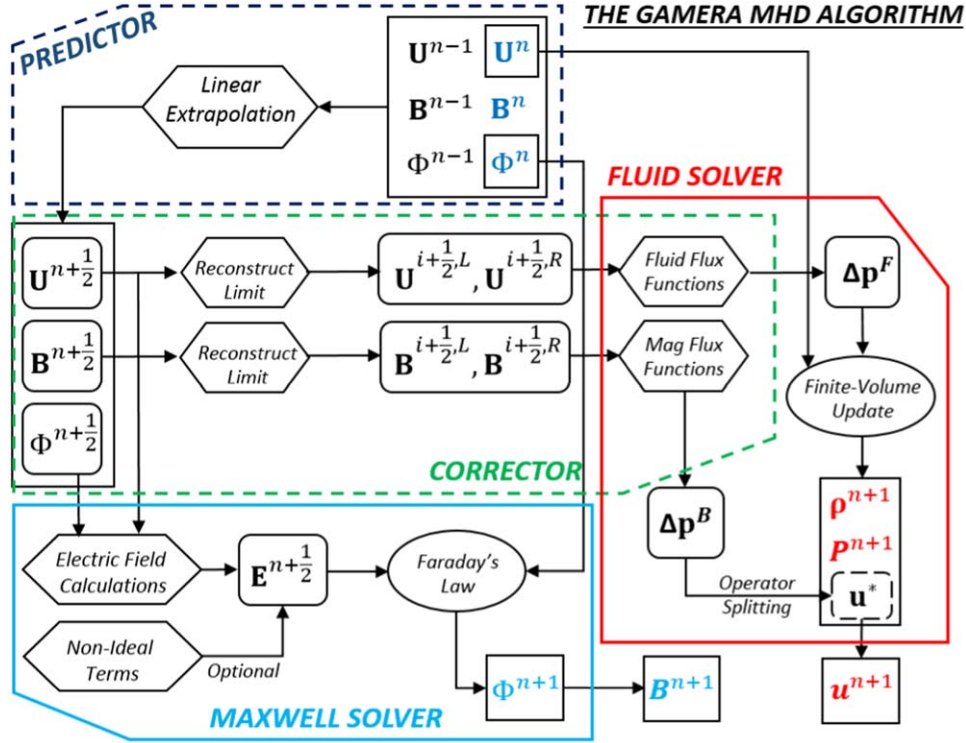


Figure 2. The main GAMERA MHD algorithm, including the predictor step, the corrector step, the fluid solver, and the Maxwell solver.

along cell edges using a constrained transport method based on the predicted velocity and magnetic flux, which is denoted as the “Maxwell Solver” and described in Section 3.5. Extensions to multistage time-stepping methods such as Runge–Kutta are possible based on the existing algorithms of the GAMERA code with appropriate modifications.

3.2.1. The Predictor Step

Assuming Q^{n-1} is a state variable at time step $n-1$, Q^n is the corresponding state variable at time step n . The predictor step for $Q^{n+\frac{1}{2}}$ at $n+\frac{1}{2}$ is computed using the following linear extrapolation in time:

$$Q^{n+\frac{1}{2}} = Q^n + \frac{\Delta t^n}{2\Delta t^{n-1}}(Q^n - Q^{n-1}), \quad (21)$$

where Δt^{n-1} and Δt^n are the time steps at $t = n-1$ and $t = n$, respectively. In the GAMERA code, the predictor steps are applied to the primitive state variables U^P (for fluid flux/stress calculations), the cell-centered Cartesian magnetic field components $B^{x,y,z}$ (for magnetic stress calculations), and the face magnetic fluxes $\Phi^{\mu,\nu,\zeta}$ (for electric-field calculations), respectively. Once the predictor step is done, the following corrector step is applied to the discrete form of Equation (13) for evolving the fluid part of the MHD equations:

$$\begin{aligned} U^{V,n+1} &= U^{V,n} + \Delta t^n \cdot F(U^{P,n+\frac{1}{2}}) \\ &\quad + \Delta t^n \cdot M(U^{P,n+\frac{1}{2}}, B^{n+\frac{1}{2}}) \\ &= U^{V,n} + \Delta U_F + \Delta U_B, \end{aligned} \quad (22)$$

where $F(U^{P,n+\frac{1}{2}})$ is a fluid flux calculation using the primitive variables $U^{P,n+\frac{1}{2}}$ at the half-time step, $M(U^{P,n+\frac{1}{2}}, B^{n+\frac{1}{2}})$ is the magnetic stress calculation using both $U^{P,n+\frac{1}{2}}$ and $B^{n+\frac{1}{2}}$, ΔU_F

is the volume-integrated changes in mass, momentum, and energy from the fluid flux/forces, and ΔU_B are the corresponding changes originating from the magnetic forces. The detailed algorithms for computing the ΔU_F and ΔU_B terms are discussed in Section 3.4. For the integral form of Faraday’s law (16), the corrector step follows

$$\Phi^{n+1} = \Phi^n + \Delta t^n \oint E^{n+\frac{1}{2}}(\Phi^{n+\frac{1}{2}}, U^{P,n+\frac{1}{2}}) \cdot dl, \quad (23)$$

where $E^{n+\frac{1}{2}}$ is the edge electric field calculated using the predicted variables $\Phi^{n+\frac{1}{2}}$ and $U^{P,n+\frac{1}{2}}$. The details of the electric field and magnetic flux evolution computations are described in Section 3.5.

This second-order Adams–Bashforth time-stepping scheme is simple and robust with the presence of nonlinear limiters with the Total variation diminishing (TVD) property (Lyon et al. 2004). The main advantage of using the Adams–Bashforth scheme is the small amount of memory and small number of computations needed for the predictor step. Thus, the second-order Adams–Bashforth method is chosen as the default method in the GAMERA code. Similarly, a third-order Adams–Bashforth scheme for the predictor is written as

$$Q^{n+\frac{1}{2}} = \frac{15}{8}Q^n - \frac{5}{4}Q^{n-1} + \frac{3}{8}Q^{n-2}, \quad (24)$$

which has slightly better stability properties since the leading term in the temporal truncation error is a fourth-order derivative term. However, the third-order Adams–Bashforth scheme requires more memory compared to the second-order method, since the two previous time steps of the state vector Q^{n-1} and Q^{n-2} are involved in the calculation of the predictor, assuming $\Delta t_{n-2} = \Delta t_{n-1} = \Delta t_n$.

Since the default Adams–Bashforth scheme is explicit in time, the time step Δt is determined by the Courant–Friedrichs–Levy

(CFL) condition. The explicit time step is calculated as

$$\Delta t = N_{\text{CFL}} \cdot \text{MIN} \left(\frac{\langle \Delta x \rangle}{v} \right), \quad (25)$$

where N_{CFL} is the Courant Number between 0 and 1, and $\langle \Delta x \rangle$ is an effective minimum “length” of a hexahedron cell as shown in Figure 1. In a computational cell indexed as (i, j, k) , the corresponding minimum cell length $\langle \Delta x \rangle$ is approximated as the cell volume $V_{i,j,k}$ divided by the maximum face area:

$$\langle \Delta x \rangle_{i,j,k} = \frac{V_{i,j,k}}{\text{MAX}(A_{i\pm\frac{1}{2},j,k}^\mu, A_{i,j\pm\frac{1}{2},k}^\nu, A_{i,j,k\pm\frac{1}{2}}^\zeta)}, \quad (26)$$

and v is the maximum wave speed within the computational cell. For the ideal MHD equations, the maximum wave speed is calculated as

$$v = |u| + \sqrt{V_A^2 + C_S^2}, \quad (27)$$

where $|u|$ is the magnitude of the plasma bulk velocity, and $\sqrt{V_A^2 + C_S^2}$ is the magnetosonic speed calculated using the Alfvén speed V_A and the plasma sound speed C_S within cell (i, j, k) . The Courant number N_{CFL} used in the default GAMERA solver is a function of numerical diffusion introduced in the default nonlinear limiter (Hain 1987), which is described in Appendix C.

3.2.2. The Corrector Step

In the corrector step, we use the “*Partial Interface Method*” (PIM) described by Lyon et al. (2004) in the LFM global magnetospheric model for handling the system of fluid equations in multidimensional, non-orthogonal geometries. The PIM provides a general framework that has the advantage of being readily adaptable to arbitrary high-order spatial reconstruction methods as well as underlying numerical flux functions. The default reconstruction algorithms and flux functions used in the GAMERA code are described in the next section.

The PIM algorithm loops over each curvilinear direction and is split into the following steps:

Step 1. Start from the predictor $U_i^{n+\frac{1}{2}}$ and $B_i^{n+\frac{1}{2}}$ in the μ -direction;

Step 2. Calculate provisional values $U_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ and $B_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ at interfaces;

Step 3. Split the provisional values into left- and right-interface states $(U_{i+\frac{1}{2}}^L, U_{i+\frac{1}{2}}^R)$ and $(B_{i+\frac{1}{2}}^L, B_{i+\frac{1}{2}}^R)$;

Step 4. Evaluate numerical flux $F(\cdot)$ and $M(\cdot)$ at cell interfaces using left and right states;

Step 5. Integrate numerical flux F and M through μ -interfaces and add to ΔU_F and ΔU_B ;

Step 6. Repeat Steps 2–5 in the ν - and ζ -directions.

The algorithm that computes ΔU_F and ΔU_B based on the predictor state is summarized in Figure 3, where Steps 2 and 3 are basically in the same vein as a typical one-dimensional TVD scheme for interface states. Note that in the PIM, the reconstruction algorithm is essentially one-dimensional; that is, the provisional values at the cell interfaces are reconstructed along each curvilinear grid coordinate μ , ν , η , and no multidimensional corrections are applied to modify the

provisional value $U_{i+\frac{1}{2}}^{n+\frac{1}{2}}$. As a consequence, the corresponding left state $(U_{i+\frac{1}{2}}^L)$ and right state $(U_{i+\frac{1}{2}}^R)$ at the cell interface $i + \frac{1}{2}$ are computed in a one-dimensional stencil along the curvilinear direction. In Step 4, the default flux functions used in the PIM for both the fluid and magnetic terms are centered flux functions with the following form:

$$F_{i+\frac{1}{2}}(U_{i+\frac{1}{2}}^L, U_{i+\frac{1}{2}}^R) = F_{\text{GK}}(U_{i+\frac{1}{2}}^L) + F_{\text{GK}}(U_{i+\frac{1}{2}}^R) \quad (28)$$

$$\begin{aligned} M_{i+\frac{1}{2}}(U_{i+\frac{1}{2}}^L, U_{i+\frac{1}{2}}^R; B_{i+\frac{1}{2}}^L, B_{i+\frac{1}{2}}^R) \\ = F_{\text{GK}}^B(U_{i+\frac{1}{2}}^L, B_{i+\frac{1}{2}}^L) + F_{\text{GK}}^B(U_{i+\frac{1}{2}}^R, B_{i+\frac{1}{2}}^R), \end{aligned} \quad (29)$$

where $F_{i+\frac{1}{2}}(\cdot)$ denotes the fluid flux terms at the cell interface, $M_{i+\frac{1}{2}}(\cdot)$ denotes the magnetic stress terms at the cell interface. $F_{\text{GK}}(\cdot)$ is a gas-kinetic flux function (described in Section 3.4) for computing the fluid flux using $U_{i+\frac{1}{2}}^L$ and $U_{i+\frac{1}{2}}^R$, and $F_{\text{GK}}^B(\cdot)$ is the corresponding magneto-gas-kinetic flux function for the Lorentz terms. The sum of two gas-kinetic integrals $F_{\text{GK}}(U_{i+\frac{1}{2}}^L)$ and $F_{\text{GK}}(U_{i+\frac{1}{2}}^R)$ gives the numerical flux through the interface

$i + \frac{1}{2}$. If we choose $U_{i+\frac{1}{2}}^L = U_i$ and $U_{i+\frac{1}{2}}^R = U_{i+1}$, the scheme becomes a robust but overly diffusive first-order gas-kinetic flux scheme developed by Croisille et al. (1995). Note that if $U^L = U^R = U$, the flux function (28) returns $F_{i+\frac{1}{2}}(U, U) = F(U^C)$ as defined in Equation (6), which is the exact form of the fluid part of the ideal MHD equations. It is only when $U^L \neq U^R$ that any numerical dissipation is introduced, although the amount of numerical dissipation introduced by the gas-kinetic scheme $F_{\text{GK}}(\cdot)$ is implicit. The numerical diffusion of that scheme is not as obvious as a first-order Rusanov scheme and has no simple description like that of Rusanov. In the case of PIM, many different numerical flux functions can also be used to replace the default gas-kinetic scheme. For example, a global heliosphere implementation of the LFM code used the HLL function for solving the MHD equations (McNutt et al. 1999). Note that the default GAMERA MHD solver uses a seventh-order reconstruction method to calculate $U_{i+\frac{1}{2}}^L$ and $U_{i+\frac{1}{2}}^R$. With such high-order reconstruction schemes, standard MHD test simulations suggest that the numerical solutions are not sensitive to the choice of the low-order flux function used in the PIM.

In Step 5, the surface integral evaluations for the mass and energy flux are straightforward since the corresponding directions of mass flux and energy flux are normal to the cell interfaces. The face integrals for momentum flux calculations in ΔU_F require coordinate transform from the local face-normal coordinate system (defined in Section 3.4.3) to the base Cartesian system in order to evolve the Cartesian components of the plasma momentum vector. This coordinate transformation enables the fluid solver to track the Cartesian components of the plasma momentum without the requirement of grid orthogonality.

When ΔU_F and ΔU_B are calculated using the PIM algorithm, an operator splitting technique is used to evolve the fluid part of the MHD Equations (1)–(3). This operator splitting technique enables us to solve for the plasma pressure P without implementing the $u \cdot \nabla \cdot \left(\bar{I} \frac{B^2}{2} - BB \right)$ term explicitly in the plasma energy equation. The algorithm shown in Figure 2 illustrates the process of updating the plasma state vector using

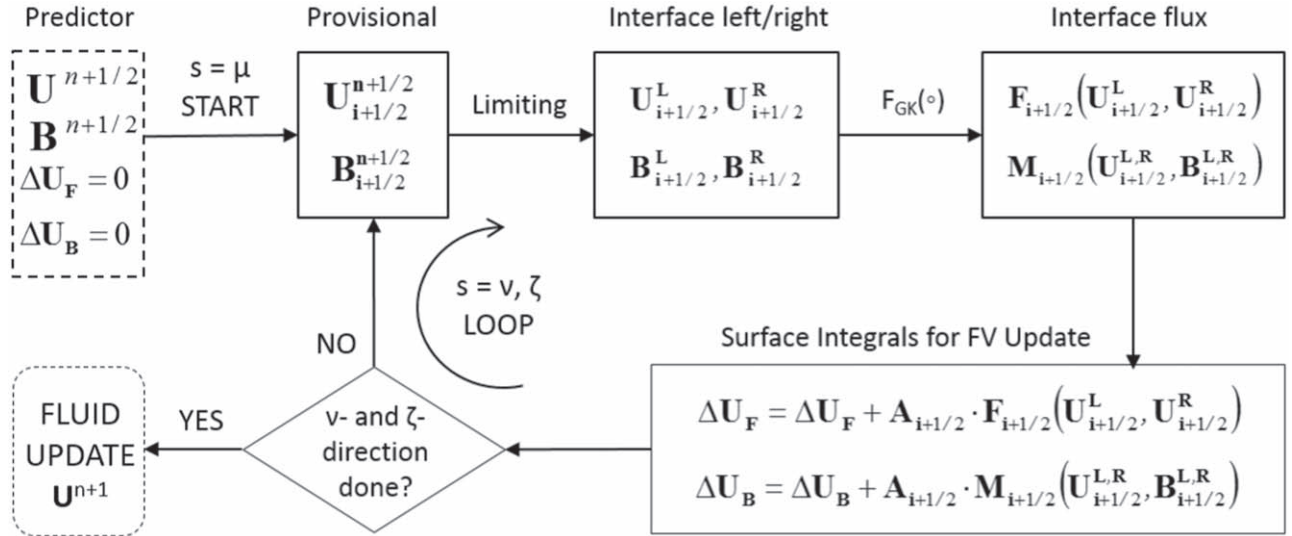


Figure 3. The Partial Interface Method for the corrector step.

the operator splitting technique described above. The first step of the operator splitting is solving the ideal gas dynamics equations using ΔU_F only:

$$\mathbf{U}^{V,n+1} = \mathbf{U}^{V,n} + \Delta \mathbf{U}_F, \quad (30)$$

and getting ρ^{n+1} and P^{n+1} with an intermediate velocity update \mathbf{u}^* , then applying the Lorentz force terms $\Delta \mathbf{U}_B$ to the intermediate velocity \mathbf{u}^* for the final update of velocity \mathbf{u}^{n+1} :

$$\mathbf{u}^{n+1} = \mathbf{u}^* + \Delta t \frac{\Delta \mathbf{U}_B}{\rho^{n+1}}. \quad (31)$$

The operator splitting technique is implemented by separating the fluid force terms $\nabla \cdot (\rho \mathbf{u} \mathbf{u} + \mathbf{I}P)$ and the magnetic force terms $\nabla \cdot (\mathbf{I} \frac{B^2}{2} - \mathbf{B} \mathbf{B})$ in the momentum equations; thus, the plasma pressure P can be solved directly from the ideal gas dynamics part of the MHD Equations (1)–(3) before applying the Lorentz force $\nabla \cdot (\mathbf{I} \frac{B^2}{2} - \mathbf{B} \mathbf{B})$ in the momentum equations, and the $\mathbf{u} \cdot \nabla \cdot (\mathbf{I} \frac{B^2}{2} - \mathbf{B} \mathbf{B})$ term is no longer needed in solving the thermal pressure P from the plasma energy equation at each time step. The operator splitting technique simplifies the implementation of the semi-relativistic (Boris) correction (Boris 1970) and the multi-fluid extension (Brambles et al. 2011) for planetary magnetosphere simulations significantly.

3.3. Calculation of Interface States

In this section, we describe the default algorithms for computing the left-state variables (\mathbf{U}^L) and right-state variables (\mathbf{U}^R) at cell interfaces for flux evaluations in the PIM framework shown in Figure 3. These interface values are computed through two consecutive steps: (1) a high-order reconstruction step, and (2) a limiting step. In the reconstruction step, the high-order reconstruction is performed on the volume-integrated state variables \mathbf{U}^V (mass, momentum, and plasma energy) in order to incorporate the variations in grid geometry. After the reconstruction step, the reconstructed \mathbf{U}^V at cell interfaces are then converted to the conservative densities \mathbf{U}^C with the geometry variations removed. In the limiting step, the reconstructed \mathbf{U}^C at the cell interface is first converted to the primitive state \mathbf{U}^P (including density, velocity, pressure, and

magnetic fields). These interface primitive variables at cell interfaces are then split into the left and right states using the nonlinear limiter algorithm. In the GAMERA code, the reconstruction and the limiting modules are implemented separately in order to make the choice of numerical schemes flexible. In the following sections, we introduce the high-order reconstruction schemes and the Partial Donor Cell Method (PDM) limiter implemented as the default in the MHD solver.

3.3.1. High-order Reconstruction

The interface values used in the GAMERA code for flux calculations are high-order approximations of the primitive variables \mathbf{U}^P and magnetic fields \mathbf{B}^{xyz} at the cell interfaces. To incorporate spatial variations originating from curvilinear geometries, the high-order reconstruction scheme operates on the volume-integrated fluid variables \mathbf{U}^V , $\mathbf{V} \cdot \mathbf{B}^{xyz}$ defined at the cell centers and $\Phi^{\mu\nu\zeta}$ at the cell faces. In the PIM framework, the reconstruction process for computing interface values is one-dimensional along each curvilinear direction, without multidimensional corrections, which simplifies the calculation in non-orthogonal geometries. For example, in order to estimate high-order approximations for interface states $\mathbf{U}_{i+\frac{1}{2}}^V$ along the μ -direction, first define a one-dimensional integral function $G(x_\mu)$ along the μ -direction as

$$G(x_\mu) = \int_{-\infty}^{x_\mu} \mathbf{U}^V(\xi) d\xi, \quad (32)$$

where \mathbf{U}^V is the vector form of the volume-integrated conservative variables, x_μ is the curvilinear spatial coordinate along the μ -direction in the computational space as shown in Figure 4. Therefore, the conserved quantity $\mathbf{U}_{i+\frac{1}{2}}^V$ at a cell interface $i + \frac{1}{2}$ along the μ -direction is calculated as the first derivative of $G(x_\mu)$ evaluated at $x_\mu = \frac{1}{2}$:

$$\mathbf{U}_{i+\frac{1}{2}}^V = \left. \frac{\partial}{\partial x_\mu} G(x_\mu) \right|_{x_\mu=i+\frac{1}{2}}. \quad (33)$$

The order of the reconstruction method to obtain interface values $\mathbf{U}_{i+\frac{1}{2}}^V$ depends on how accurately the first derivative term

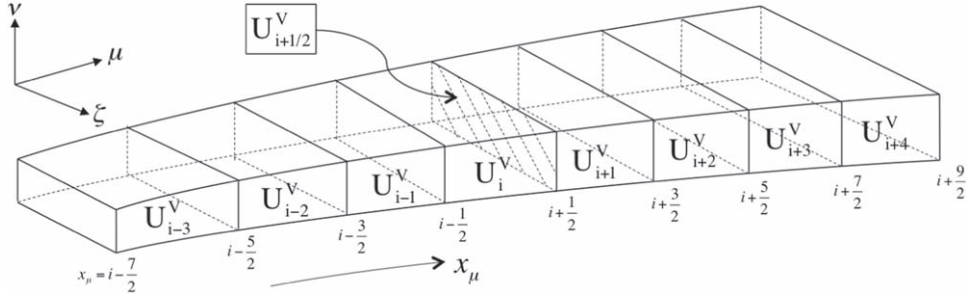


Figure 4. A one-dimensional, high-order reconstruction stencil for computing $U_{i+1/2}^V$ in the μ -direction.

in Equation (33) is approximated. An example of an eight-cell reconstruction stencil along the μ -direction is shown in Figure 4. If applying a second-order centered approximation for the numerical derivative $\frac{\partial}{\partial x_\mu} G(x_\mu)$ at the interface $i + \frac{1}{2}$ as an example, the rhs of Equation (33) is calculated as

$$U_{i+1/2}^V = \frac{\partial}{\partial x_\mu} G(x_\mu) \Big|_{i+1/2} = \frac{G(i + \frac{3}{2}) - G(i - \frac{1}{2})}{(i + \frac{3}{2}) - (i - \frac{1}{2})} + \mathcal{O}(\Delta x_\mu^2) \approx \frac{U_{i+1}^V + U_i^V}{2}. \quad (34)$$

Note that the integral function $G(x_\mu)$ is defined in a computational space between $x_\mu = i - \frac{7}{2}$ and $x_\mu = i + \frac{9}{2}$ since the information of the cell volume is already included in $G(x_\mu)$. After $U_{i+1/2}^V$ is computed, the conserved state variables $U_{i+1/2}^C$ at interface $i + 1/2$ are calculated as

$$U_{i+1/2}^C = \frac{U_{i+1/2}^V}{V_{i+1/2}} = \frac{1}{2V_{i+1/2}}(U_{i+1}^V + U_i^V), \quad (35)$$

where $V_{i+1/2}$ is an estimation of “cell volume” at the corresponding cell interface $i + \frac{1}{2}$ assuming the cell volume V_i varies smoothly in the μ -direction. A second-order centered approximation of the interface volume for the second-order interface reconstruction in (35) is calculated as

$$V_{i+1/2} = \frac{1}{2}(V_{i+1} + V_i). \quad (36)$$

Once $U_{i+1/2}^C$ is calculated, the primitive state variable $U_{i+1/2}^P$ is then computed for the limiting step described in the next section. The interface magnetic field $B_{i+1/2}$ is calculated using the same algorithm by replacing U^V with $V_{i,j,k} \cdot B_{i,j,k}$. In the original LFM code, the reconstruction module uses the eight-cell stencil shown in Figure 4 with the following eighth-order centered reconstruction for computing high-order estimations of interface values:

$$U_{i+1/2}^C = \frac{1}{V_{i+1/2}^{8th}} \left(-\frac{3}{840}U_{i-3}^V + \frac{29}{840}U_{i-2}^V - \frac{139}{840}U_{i-1}^V + \frac{533}{840}U_i^V + \frac{533}{840}U_{i+1}^V - \frac{139}{840}U_{i+2}^V + \frac{29}{840}U_{i+3}^V - \frac{3}{840}U_{i+4}^V \right), \quad (37)$$

where $V_{i+1/2}^{8th}$ is an eighth-order approximation of the interface volume calculated as

$$V_{i+1/2}^{8th} = -\frac{3}{840}V_{i-3} + \frac{29}{840}V_{i-2} - \frac{139}{840}V_{i-1} + \frac{533}{840}V_i + \frac{533}{840}V_{i+1} - \frac{139}{840}V_{i+2} + \frac{29}{840}V_{i+3} - \frac{3}{840}V_{i+4}. \quad (38)$$

This eighth-order centered reconstruction scheme is chosen as the default method for computing the interface state variables in the LFM code based on its high-resolving power of contact discontinuities according to Lyon et al. (2004).

In fact, the reconstruction method for estimating a high-order interface value is not necessarily centered. High-order upwind reconstruction schemes are also implemented in the GAMERA code. Using the second-order centered reconstruction as an example, the interface variable is reconstructed as

$$U_{i+1/2}^V = \frac{1}{2}(U_i^V + U_{i+1}^V). \quad (39)$$

Combining the above second-order approximation for $U_{i+1/2}^V$ with a second-order centered approximation for the first derivative of U^V evaluated at $i + \frac{1}{2}$, the centered reconstruction becomes the first-order upwind method (LeVeque 2002):

$$U_{i+1/2}^V = U_{i+1/2}^{V|2nd} - \frac{1}{2} \frac{\partial}{\partial x} U_{i+1/2}^{V|2nd} = \frac{1}{2}(U_i^V + U_{i+1}^V) - \frac{1}{2}(U_{i+1}^V - U_i^V) = U_i^V. \quad (40)$$

Equation (40) suggests that upwind reconstruction with an order of $n - 1$ can be derived from a n th-order (where n is even) centered stencil by canceling the outermost cell in the downwind direction using a numerical first derivative approximated to n th-order accuracy at the cell interface. For example, combining the eighth-order centered interpolation for $U_{i+1/2}^V$ with an eighth-order centered approximation of $\frac{1}{2} \frac{\partial U^V}{\partial x_\mu}$, the following seventh-order upwind reconstruction method is obtained:

$$U_{i+1/2}^V = -\frac{1}{40}U_{i-3}^V + \frac{5}{84}U_{i-2}^V - \frac{101}{420}U_{i-1}^V + \frac{319}{420}U_i^V + \frac{107}{210}U_{i+1}^V - \frac{19}{210}U_{i+2}^V + \frac{1}{105}U_{i+3}^V. \quad (41)$$

In the above seventh-order reconstruction, the rightmost cell $i + 4$ shown in Figure 4 is removed from the reconstruction stencil, which shifts the reconstruction toward left (the upwind

Table 3
Centered and Upwind Reconstruction Coefficients for $U_{i+\frac{1}{2}}$, up to 12th Order

	U_{i-5}	U_{i-4}	U_{i-3}	U_{i-2}	U_{i-1}	U_i	U_{i+1}	U_{i+2}	U_{i+3}	U_{i+4}	U_{i+5}	U_{i+6}
1st						1						
2nd						$\frac{1}{2}$	$\frac{1}{2}$					
3th					$-\frac{1}{6}$	$\frac{5}{6}$	$\frac{1}{3}$					
4th					$-\frac{1}{12}$	$\frac{7}{12}$	$\frac{1}{6}$	$-\frac{1}{12}$				
5th				$\frac{1}{30}$	$-\frac{13}{60}$	$\frac{47}{60}$	$\frac{9}{20}$	$-\frac{1}{20}$				
6th				$\frac{1}{60}$	$-\frac{2}{15}$	$\frac{37}{60}$	$\frac{37}{60}$	$-\frac{2}{15}$	$\frac{1}{60}$			
7th			$-\frac{1}{40}$	$\frac{5}{84}$	$-\frac{101}{420}$	$\frac{319}{420}$	$\frac{107}{210}$	$-\frac{19}{105}$	$\frac{1}{105}$			
8th			$-\frac{3}{840}$	$\frac{29}{840}$	$-\frac{139}{840}$	$\frac{533}{840}$	$\frac{533}{840}$	$-\frac{139}{840}$	$\frac{29}{840}$	$-\frac{3}{840}$		
9th		$\frac{1}{630}$	$-\frac{41}{2520}$	$\frac{840}{2520}$	$-\frac{641}{2520}$	$\frac{1879}{2520}$	$\frac{275}{504}$	$-\frac{61}{473}$	$\frac{11}{504}$	$\frac{840}{504}$	$-\frac{1}{23}$	
10th		$\frac{1}{1260}$	$-\frac{23}{2520}$	$\frac{127}{2520}$	$-\frac{473}{2520}$	$\frac{747}{2520}$	$\frac{504}{1157}$	$-\frac{473}{1157}$	$\frac{127}{2520}$	$\frac{504}{2520}$	$\frac{1}{1260}$	
11th	$\frac{1}{2772}$	$\frac{13860}{67}$	$-\frac{58}{2287}$	$\frac{371}{3960}$	$-\frac{93}{353}$	$\frac{260}{353}$	$\frac{420}{737}$	$-\frac{133}{921}$	$\frac{339}{9923}$	$\frac{17}{3080}$	$\frac{1}{2310}$	
12th	$-\frac{1}{5544}$	$\frac{27720}{27720}$	$-\frac{107}{6930}$	$\frac{443}{6930}$	$-\frac{529}{2594}$	$\frac{356}{545}$	$\frac{529}{545}$	$-\frac{529}{2594}$	$\frac{443}{6930}$	$\frac{107}{6930}$	$\frac{67}{27720}$	$-\frac{1}{5544}$

Note. The upwind coefficients are for the left-state interface values with the stencil shifted toward the left.

direction). Reconstruction coefficients for left interface states from the first order to 12th order are listed in Table 3. The right interface states are calculated using the same reconstruction coefficients by switching the upwind direction of a reconstruction stencil. For the centered reconstruction methods (e.g., eighth-order), the left and right states are computed using the same eight-cell stencil and coefficients due to symmetry; for the upwind reconstruction methods, (e.g., seventh-order), the left and right states are computed using two different seven-cell stencils shifted toward the left and right sides of the interface, respectively. In the GAMERA code, the solver uses the seventh-order upwind reconstruction scheme as the default choice, while the original LFM MHD kernel uses the eighth-order centered reconstruction. The eighth-order reconstruction scheme has a leading truncation error of a dispersive ninth-order derivative; thus, a sudden change in gradient (involving short wavelengths) may excite spurious unphysical oscillations due to the ninth-derivative dispersion term in the truncation error. However, in the seventh-order reconstruction scheme, the leading truncation-error term is an eighth-order spatial derivative, which is dissipative instead of dispersive. This is an important improvement in the original LFM algorithm. Numerical experience over the past decade or more has shown that for numerical solutions of convection-dominated fluid flows, the leading truncation error in a convection algorithm should be dissipative (an even spatial derivative term) rather than dispersive (an odd spatial derivative term), which should also be of a higher order than the modeled physical diffusive terms (if any; Leonard & Niknafs 1991). Thus, the odd-order (order ≥ 3) reconstruction schemes are more natural choices, satisfying the criteria of (1) dissipative truncation-error terms and (2) higher-order than physical diffusion terms. Thus, the GAMERA code uses a seventh-order reconstruction method as the default choice for upwind spatial reconstruction, and the eighth-order reconstruction method is chosen as the default centered method for reference. The reason for choosing such high-order reconstruction methods as the default in the MHD solver is explained in Appendix D, which is based on a simple measure of optimizing between low numerical diffusion and

high computing efficiency as well as the need to resolve both physical and grid structure.

3.3.2. Partial Donor Cell Method

When discontinuities occur within a reconstruction stencil, a problem arises in using the high-order reconstruction schemes, since spurious undershoots or overshoots may occur near discontinuities. To avoid this problem, nonlinear switchers are used to “correct” the high-order reconstructed values on both sides of the interface when necessary. In the limiting module, the PDM limiter developed by Hain (1987) is implemented as the default choice, which does not depend on the numerical order of interface reconstruction. The basic idea of the PDM limiter is that the algorithm monitors sharp discontinuities; if a sharp discontinuity is identified, a “limited” value is used in order to keep the solution from overshoots/undershoots; otherwise, a high-order approximation is always chosen for the interface state to provide a more accurate estimation. Details about the PDM limiter can be found in Hain (1987). Here, we provide a simplified description of how the PDM limiter works adapted from Huba (2003).

Consider a density structure being advected at a constant velocity V toward the right direction as shown in Figure 5. The PDM limiter determines whether the high-order estimation of the left-state value $\rho_{L_{i+\frac{1}{2}}}^{\text{HO}}$ at interface $i + 1/2$ needs to be “limited” for monotonicity preserving purpose. In Figure 5, ρ_{i-1} is the density in cell $i - 1$, ρ_i is the density in cell i , $\rho_{i+1/2}^{\text{HO}}$ is the high-order reconstructed density at cell interface $i + 1/2$, and $\rho_{i+1/2}^{\text{PDM}}$ is the limited value (or the PDM value) of the interface density on the left side, which will now be determined.

Assuming the density structure is advanced for one time step Δt , it advects toward the right by a finite distance $V\Delta t$. After one advection step, the amount of mass entering cell i from cell $i - 1$ is $\rho_{i-1}V\Delta t$, which is illustrated by the orange shaded area in Figure 5; the amount of mass leaving cell i and entering cell $i + 1$ is $\rho_{i+1/2}^{\text{PDM}}V\Delta t$, assuming that the left state is the PDM value that guarantees no overshooting/undershooting, which is illustrated by the green shaded area in Figure 5. Therefore, the total density change in cell i after one time step is calculated as

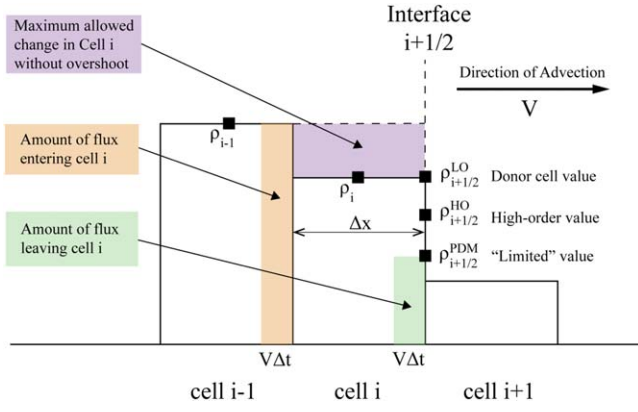


Figure 5. The calculation of the PDM value on the left side of the interface $i + \frac{1}{2}$, using the advection of density in a one-dimensional stencil toward the right. The figure is adapted from Huba (2003).

$(\rho_{i-1} - \rho_{i+1/2}^{\text{PDM}})V\Delta t$. Since the maximum density increase allowed in cell i without spurious overshoot is $(\rho_{i-1} - \rho_i)\Delta x$, which is denoted by the purple area, the left state of the PDM value at cell interface $i + \frac{1}{2}$ is derived by balancing the two quantities

$$(\rho_{i-1} - \rho_{i+1/2}^{\text{PDM}})V\Delta t = (\rho_{i-1} - \rho_i)\Delta x, \quad (42)$$

which gives the PDM value

$$\rho_{L_{i+\frac{1}{2}}}^{\text{PDM}} = \frac{1}{\epsilon}\rho_i - \left(1 - \frac{1}{\epsilon}\right)\rho_{i-1}, \quad (43)$$

where $\epsilon = V\Delta t/\Delta x$ is in the range between 0 and 1. By balancing the density entering and leaving cell i , it is clear that the $\rho_{L_{i+\frac{1}{2}}}^{\text{PDM}}$ is the minimum value of the left-state density allowed to leave cell i without spurious overshoot. If the left-state density is lower than this value, spurious density overshoot occurs within cell i . By choosing the $\rho_{L_{i+\frac{1}{2}}}^{\text{PDM}}$ as the left state, the density on the left side of the interface is “limited.” Note that if $\epsilon = 1$, then $\rho_{L_{i+\frac{1}{2}}}^{\text{PDM}} = \rho_i$, which is the first-order Donor Cell method with an excessive amount of numerical diffusion for nonlinear problems. By choosing $\epsilon < 1$, the above scheme takes a portion of the Donor Cell solution as the limited value and reduces the amount of numerical diffusion significantly.

To achieve a high-order approximation for the left state, it is not necessary to use the PDM value all of the time. As shown in Figure 5, there are three density values to choose as the left state at interface $i + \frac{1}{2}$: ρ_i (first order), $\rho_{i+1/2}^{\text{HO}}$ (seventh-order reconstruction as the default) and $\rho_{i+1/2}^{\text{PDM}}$ (the limited value). The rationale for choosing the final left state value is as follows. In general, one would want to use the high-order reconstructed value because it provides the best accuracy for estimating the left state. Since $\rho_{i+1/2}^{\text{PDM}}$ is the minimum amount of density allowed for a left state leaving cell i , as long as $\rho_{i+1/2}^{\text{PDM}} < \rho_{i+1/2}^{\text{HO}} < \rho_i$, the high-order value $\rho_{i+1/2}^{\text{HO}}$ is the correct choice for the left state. However, when $\rho_{i+1/2}^{\text{HO}} < \rho_{i+1/2}^{\text{PDM}} < \rho_i$, the PDM value $\rho_{i+1/2}^{\text{PDM}}$ must be chosen as the left value in order to avoid overshoots. In other words, the left state is always chosen as the median value of the three interface values: ρ_i ,

$\rho_{i+1/2}^{\text{HO}}$, and $\rho_{i+1/2}^{\text{PDM}}$. The PDM limiter is based on balancing the amount of flux moving through a controlled volume rather than on constraining the slope of the underlying reconstruction polynomial for computing interface values, which is similar to the “universal Limiter” developed by Leonard & Niknafs (1991). Thus, the interface value $\rho_{i+1/2}^{\text{HO}}$ can be replaced by arbitrary high-order reconstructed values. The right-state density is derived in the same way through the advection of a density structure toward the left. The performance of the PDM limiter combined with different orders of reconstruction methods is demonstrated in Appendix D based on quantitative comparisons of 1D linear advection test results.

For the linear advection equation, it is straightforward to show that the PDM limiter is TVD (Hain 1987); thus, it tends to “clip” local extrema as typical TVD limiters do. An optional non-clipping algorithm is implemented in the limiting module to preserve local extrema for simulations that are sensitive to the numerical errors introduced due to clipping of local extrema, e.g., the Harris current sheet equilibrium simulation. A simple non-clipping algorithm developed by Leonard & Niknafs (1991) is implemented in the limiting process in order to distinguish between artificial numerical peaks and true physical extrema. Figure 6 shows the difference between a physical extremum and a numerical extremum within one reconstruction stencil centered at cell i . If a local extremum is associated with short-wavelength oscillations, the limited value is chosen for the interface state; however, if the curvature of a peak follows the shape of a local extremum, the unlimited high-order value is chosen to avoid clipping. Using the stencil for calculating the left state $U_{i+\frac{1}{2}}$ at the interface $i + \frac{1}{2}$ is shown in Figure 6 as an example. The calculation of the local extrema indicator (LEI) is illustrated in Figure 6, by first computing the differences between each pair of consecutive cell center values:

$$\begin{aligned} D_1 &= U_{i-1} - U_{i-2}, D_2 = U_i - U_{i-1}, \\ D_3 &= U_{i+1} - U_i, D_4 = U_{i+2} - U_{i+1}. \end{aligned} \quad (44)$$

Assuming the existence of one local maximum centered in cell i as shown in the left panel of Figure 6 (a minimum requires a reversal of some of the subsequent inequalities), both D_1 and D_2 are positive and D_3 and D_4 are negative. Using a simple constraint for such a local maximum with decreasing gradient toward the peak, $|D_1| > |D_2|$ and $|D_3| < |D_4|$, the LEI for the left state $U_{i+\frac{1}{2}}$ in cell i is calculated as a Bool type of variable:

$$\begin{aligned} \text{LEI} &= (D_1 > 0) \& (D_2 > 0) \& (D_3 < 0) \& (D_4 < 0) \\ &\& (|D_1| > |D_2|) \& (|D_3| < |D_4|). \end{aligned} \quad (45)$$

When the non-clipping option is switched on, after the limiting step, the final interface state is selected based on the value of the LEI, if $\text{LEI} = \text{TRUE}$, the unlimited, high-order approximation (e.g., the $\rho_{i+1/2}^{\text{HO}}$) is used for the interface state regardless of the $\rho_{i+1/2}^{\text{PDM}}$ constraint; otherwise, if $\text{LEI} = \text{FALSE}$, the limited value is chosen to avoid possible spurious overshooting/undershooting.

Combining the reconstruction and the limiting step, the algorithm to compute the left- and right-state variables along one-dimensional stencils (e.g., in the μ -direction) is summarized as follows:

Step 1. Compute the volume-integrated conserved quantities in each controlled volume using $U_{i,j,k}^V = V_{i,j,k} \cdot U_{i,j,k}^C$;

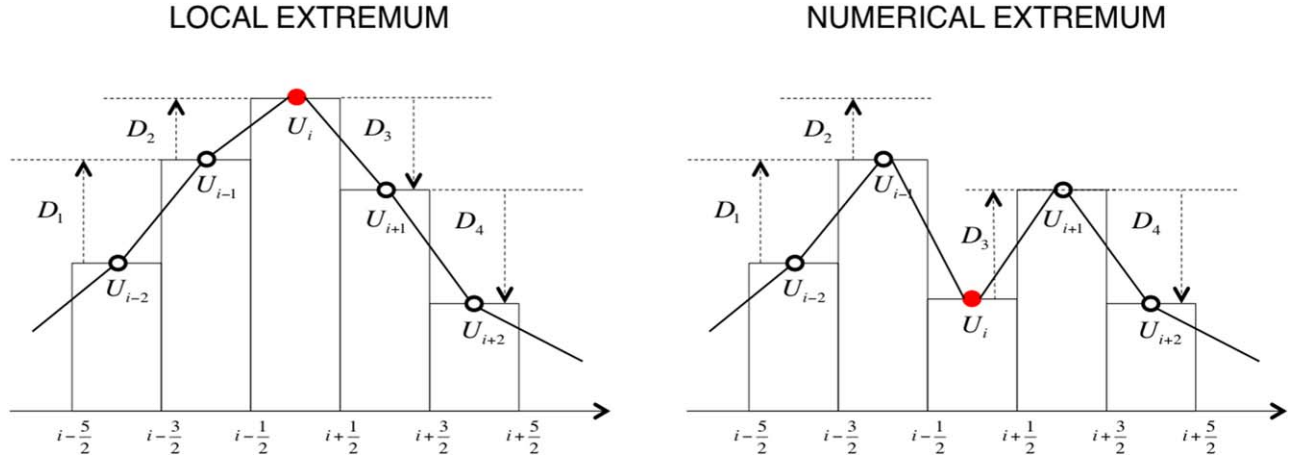


Figure 6. A local maximum vs. numerical peaks within a stencil of five computational cells.

Step 2. Compute a high-order (default seventh-order) reconstructed volume-integrated quantity $U_{i+\frac{1}{2},j,k}^V$;

Step 3. Estimate a high-order (default eighth-order) volume $V_{i+\frac{1}{2},j,k}$ at the interface location where $U_{i+\frac{1}{2},j,k}^V$ is defined;

Step 4. Compute the reconstructed densities $U_{i+\frac{1}{2},j,k}^C = U_{i+\frac{1}{2},j,k}^V / V_{i+\frac{1}{2},j,k}$, then convert to primitive variables $U_{i+\frac{1}{2},j,k}^P$;

Step 5. Split the interface primitive variable $U_{i+\frac{1}{2},j,k}^P$ into the left- and right-state variables using the PDM limiter.

Step 6 (optional). Apply the non-clipping sweep to decide whether the PDM limiter is needed or not.

3.4. Gas-kinetic Flux Functions

The default flux functions for the fluid part of the MHD equations are based on one-dimensional gas-kinetic schemes for the Euler equations in the face-normal coordinate system (detailed in Section 3.4.3). In a local, orthogonal coordinate system (x_n, x_1, x_2) with x_n as the direction normal to a cell interface and x_1, x_2 as the two corresponding orthogonal directions tangential to the cell interface, the one-dimensional mass, momentum, and plasma energy Equations (1)–(3) in this (x_n, x_1, x_2) coordinate system are written as

$$\begin{aligned} \frac{\partial}{\partial t} \begin{bmatrix} \rho \\ \rho u_{x_n} \\ \rho u_{x_1} \\ \rho u_{x_2} \\ E_p \end{bmatrix} &= - \frac{\partial}{\partial x_n} \begin{bmatrix} \rho u_{x_n} \\ \rho u_{x_n}^2 + P \\ \rho u_{x_n} u_{x_1} \\ \rho u_{x_n} u_{x_2} \\ u_{x_n} (E_p + P) \end{bmatrix} - \frac{\partial}{\partial x_n} \begin{bmatrix} 0 \\ B^2/2 - B_{x_n}^2 \\ B_{x_n} B_{x_1} \\ B_{x_n} B_{x_2} \\ 0 \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u_{x_n} \frac{\partial}{\partial x_n} (B^2/2 - B_{x_n}^2) + u_{x_1} \frac{\partial}{\partial x_1} B_{x_n} B_{x_1} + u_{x_2} \frac{\partial}{\partial x_2} B_{x_n} B_{x_2} \end{bmatrix}, \end{aligned} \quad (46)$$

where E_p is the plasma energy defined in (5), and $B^2/2$ is the magnetic energy calculated as $(B_{x_n}^2 + B_{x_1}^2 + B_{x_2}^2)/2$. The first part on the rhs of Equation (46) corresponds to the ideal gas dynamics with velocity components in the (x_n, x_1, x_2)

coordinate system, which gives the ΔU_F terms in Equation (22). The second part on the rhs of Equation (46) is the Lorentz force $\nabla \cdot (\mathbf{I} \frac{B^2}{2} - \mathbf{B}\mathbf{B})$ in the momentum in Equation (2), which gives the ΔU_B terms in Equation (22). The third part on the rhs of Equation (46) is the work done by the Lorentz force $\mathbf{u} \cdot \nabla \cdot (\mathbf{I} \frac{B^2}{2} - \mathbf{B}\mathbf{B})$ in the plasma energy in Equation (3), which is not computed explicitly for solving the plasma pressure P , due to the operator splitting technique as discussed in Section 3.2.2. The gas-hydro flux (fluid stress) and the magneto-hydro flux (magnetic stress) for the above one-dimensional MHD equations are calculated separately, which simplifies the implementation of the semi-relativistic (Boris) correction for magnetospheric simulations. The default solver uses a Maxwellian-based gas-kinetic flux function to compute the fluid fluxes terms ΔU_F and uses another Maxwellian-based magneto-gas-kinetic flux function to compute the magnetic stresses terms ΔU_B .

3.4.1. Fluid Fluxes

The default flux functions used in the solver for the fluid flux calculations are gas-kinetic schemes based on Maxwellian distributions, which consider the fluid on each side of a cell interface as microscopic distributions of noninteracting particles (Croisille et al. 1995). These particles are allowed to free-stream across the interface for one time step, then the macroscopic conserved variables including mass, momentum, and energy are derived from the zeroth, first, and second moment integral of the corresponding distribution functions on both sides of the interface. These updated macroscopic variables are then used to construct new distribution functions for the next time step.

Using the left primitive state variables $U^{P,L} = (\rho^L, \mathbf{u}^L, P^L)^T$ calculated in the reconstruction module, the distribution function $f^L(v)$ for the plasma population on the left side of the cell interface is a Maxwell-Boltzmann distribution function:

$$f^L(v) = \sqrt{\frac{\rho^L}{2\pi P^L}} e^{-\frac{\rho^L}{2P^L}(v - u_{x_n}^L)^2}. \quad (47)$$

Similarly, the corresponding distribution function $f^R(v)$ for the plasma population on the right side of the cell interface is

$$f^R(v) = \sqrt{\frac{\rho^R}{2\pi P^R}} e^{-\frac{\rho^R}{2P^R}(v-u_{x_n}^R)^2}, \quad (48)$$

where ρ^L and ρ^R are the left and right state of plasma density, P^L and P^R are the left and right state of plasma pressure, $u_{x_n}^L$ and $u_{x_n}^R$ are the left and right plasma bulk velocity in the x_n -direction normal to the cell interfaces, respectively. The calculations of $u_{x_n}^L$ and $u_{x_n}^R$ from the interface velocities \mathbf{u}^L and \mathbf{u}^R are described in the next section. For one-dimensional MHD flows described by Equation (46) with particles free-streaming in the x_n -direction, the moment integrals in this direction determine the form of flux functions. Thus, in the local (x_n, x_1, x_2) coordinate system, the mass flux, the fluid part of the momentum flux, and the energy flux across the cell interface $i + \frac{1}{2}$ are calculated by integrating the corresponding distribution functions for the left-going particles on the right side of the interface and the right-going particles on the left side of the interface as

$$\mathbf{F}_{\text{FLUID}}^n = \begin{pmatrix} F_\rho \\ F_{\rho u_{x_n}} \\ F_{\rho u_{x_1}} \\ F_{\rho u_{x_2}} \\ F_{E_p} \end{pmatrix} = \int_0^{+\infty} f^L(v) \begin{bmatrix} \rho^L v \\ \rho v^2 + P^L \\ \rho v u_{x_1}^L \\ \rho v u_{x_2}^L \\ v(E_p^L + P^L) \end{bmatrix} dv + \int_{-\infty}^0 f^R(v) \begin{bmatrix} \rho^R v \\ \rho^R v^2 + P^R \\ \rho v u_{x_1}^R \\ \rho v u_{x_2}^R \\ v(E_p^R + P^R) \end{bmatrix} dv, \quad (49)$$

where $\mathbf{F}_{\text{FLUID}}^n$ is the vector form of the fluid fluxes at interface n ($n = \mu, \nu, \zeta$), F_ρ is the mass flux, $F_{\rho u}$ is the momentum flux vector, and F_{E_p} is the plasma energy flux calculated in the (x_n, x_1, x_2) coordinates. $u_{x_1,2}^L$ and $u_{x_1,2}^R$ are the corresponding left- and right-interface *tangential* velocity components transformed into the local (x_n, x_1, x_2) coordinate system, respectively. After the fluid fluxes are calculated, the ΔU_F term for the correction step of Equation (22) is computed as

$$\Delta U_F = \Delta t \cdot \oint \mathbf{F}_{\text{FLUID}}^n \cdot d\mathbf{S} = \sum_{s=\mu,\nu,\zeta} \mathbf{F}_{\text{FLUID}}^s \cdot A_s \mathbf{n}_s, \quad (50)$$

where A_s is the face area and \mathbf{n}_s is the corresponding face-normal unit vector. If the left and right states are identical, i.e., $\mathbf{U}^L = \mathbf{U}^R \equiv \mathbf{U}$, it is straightforward to show that $f^L(v) \equiv f^R(v)$ and the rhs of one-dimensional ideal gas dynamics equations are recovered through the flux functions of Equation (49). The detailed derivations for evaluating the moment integrals in Equation (49) can be found in Xu (1999). Here, we only give the mathematical expressions of the Maxwellian-distribution-based

gas-kinetic flux scheme for computing fluid fluxes:

$$\mathbf{F}_{\text{fluid}} = \langle v^1 \rangle_+^L \begin{pmatrix} \rho_L \\ \rho_L u_{x_n}^L \\ \rho^L u_{x_1}^L \\ \rho^L u_{x_2}^L \\ E_p^L \end{pmatrix} + \begin{pmatrix} 0 \\ P^L \langle v^0 \rangle_+^L \\ 0 \\ 0 \\ P^L \langle v^1 \rangle_+^L / 2 + P^L u_{x_n}^L \langle v^0 \rangle_+^L \end{pmatrix} + \langle v^1 \rangle_-^R \begin{pmatrix} \rho_R \\ \rho_R u_{x_n}^R \\ \rho^R u_{x_1}^R \\ \rho^R u_{x_2}^R \\ E_p^R \end{pmatrix} + \begin{pmatrix} 0 \\ P^R \langle v^0 \rangle_-^R \\ 0 \\ 0 \\ P^R \langle v^1 \rangle_-^R / 2 + P^R u_{x_n}^R \langle v^0 \rangle_-^R \end{pmatrix}. \quad (51)$$

The zeroth and first velocity moments of the left and right distribution functions used in Equations (51) are calculated as

$$\begin{aligned} \langle v^0 \rangle_+^L &= \int_0^{+\infty} v^0 f^L(v) dv = \frac{1}{2} \operatorname{erfc}(-u_{x_n}^L \sqrt{\lambda^L}), \\ \langle v^0 \rangle_-^R &= \int_{-\infty}^0 v^0 f^R(v) dv = \frac{1}{2} \operatorname{erfc}(-u_{x_n}^R \sqrt{\lambda^R}), \end{aligned} \quad (52)$$

$$\begin{aligned} \langle v^1 \rangle_+^L &= \int_0^{+\infty} v^1 f^L(v) dv = u_{x_n}^L \langle v^0 \rangle_+^L + \frac{1}{2} \frac{e^{-\lambda^L u_{x_n}^L{}^2}}{\sqrt{\pi \lambda^L}}, \\ \langle v^1 \rangle_-^R &= \int_{-\infty}^0 v^1 f^R(v) dv = u_{x_n}^R \langle v^0 \rangle_-^R + \frac{1}{2} \frac{e^{-\lambda^R u_{x_n}^R{}^2}}{\sqrt{\pi \lambda^R}}, \end{aligned} \quad (53)$$

where $\lambda^L = \rho^L / 2P^L$ and $\lambda^R = \rho^R / 2P^R$. The $\operatorname{erfc}(\cdot)$ function is the *Complementary Error Function*. It is straightforward to show that if $f^L(v) = f^R(v) \equiv f(v)$, the following relationships hold:

$$\langle v^0 \rangle_+ + \langle v^0 \rangle_- = 1, \quad (54)$$

$$\langle v^1 \rangle_+ + \langle v^1 \rangle_- = u_{x_n}. \quad (55)$$

The mass flux F_ρ and energy flux F_{E_p} in Equations (51) are in the x_n -direction normal to the cell interfaces such that

$$F_\rho = (\langle v^1 \rangle_+^L \rho_L + \langle v^1 \rangle_-^R \rho_R) \mathbf{n}_{x_n}, \quad (56)$$

$$\begin{aligned} F_{E_p} &= (\langle v^1 \rangle_+^L E_p^L + P^L \langle v^1 \rangle_+^L / 2 + P^L u_{x_n}^L \langle v^0 \rangle_+^L \\ &\quad + P^R \langle v^1 \rangle_-^R / 2 + P^R u_{x_n}^R \langle v^0 \rangle_-^R \langle v^1 \rangle_-^R E_p^R) \mathbf{n}_{x_n}. \end{aligned} \quad (57)$$

Thus, the mass and energy flux are used to compute the surface integrals in Equation (50) directly:

$$\Delta U_F|_\rho = \Delta t \sum_{s=\mu,\nu,\zeta} \mathbf{F}_\rho^s \cdot A_s \mathbf{n}_s \quad (58)$$

$$\Delta U_F|_{E_p} = \Delta t \sum_{s=\mu,\nu,\zeta} \mathbf{F}_{E_p}^s \cdot A_s \mathbf{n}_s. \quad (59)$$

However, the momentum flux vector $\mathbf{F}_{\rho u}$ calculated using Equation (51) is defined in the local (x_n, x_1, x_2) coordinate

system as

$$\begin{aligned} \mathbf{F}_{\rho\mathbf{u}} = & (\langle v^1 \rangle_+^L \rho_L \mathbf{u}_{x_n}^L + P^L \langle v^0 \rangle_+^L \\ & + \langle v^1 \rangle_-^R \rho_R \mathbf{u}_{x_n}^R + P^L \langle v^0 \rangle_-^R) \mathbf{n}_{x_n} \\ & + (\langle v^1 \rangle_+^L \rho_L \mathbf{u}_{x_1}^L + \langle v^1 \rangle_-^R \rho_R \mathbf{u}_{x_1}^R) \mathbf{n}_{x_1} \\ & + (\langle v^1 \rangle_+^L \rho_L \mathbf{u}_{x_2}^L + \langle v^1 \rangle_-^R \rho_R \mathbf{u}_{x_2}^R) \mathbf{n}_{x_2}, \end{aligned} \quad (60)$$

which cannot be used to compute the surface integrals in Equation (50) directly in order to update the base Cartesian components of the momentum (ρu_x , ρu_y , ρu_z). Thus, after evaluating the fluid fluxes in the face-normal coordinate system (x_n , x_1 , x_2) using the one-dimensional flux function form of Equation (51), the momentum flux vector $\mathbf{F}_{\rho\mathbf{u}}$ at the cell interface is then rotated back to the base Cartesian coordinate system for updating the Cartesian component of the momentum (ρu_x , ρu_y , ρu_z). Then, the changes to the plasma momentum $\Delta U_{F|\rho\mathbf{u}}$ in the base Cartesian system are computed as

$$\Delta U_{F,\rho\mathbf{u}} = \Delta t \sum_{s=\mu,\nu,\zeta} \bar{\mathbf{T}}^{-1} \cdot \mathbf{F}_{\rho\mathbf{u}}^s A_s, \quad (61)$$

where $\bar{\mathbf{T}}^{-1}$ is a matrix for transforming the momentum flux vector from the (x_n , x_1 , x_2) system to the base Cartesian system (x , y , z). The detailed calculation of the transform matrix $\bar{\mathbf{T}}^{-1}$ is described in Section 3.4.3. This rotation method for evaluating momentum fluxes in the base Cartesian coordinate system (x , y , z) does not require the orthogonality of the physical grid and works with any microscopic distribution functions in the gas-kinetic flux scheme.

The gas-hydro distribution function for the fluid flux calculations does not depend on the local Alfvén speed; therefore, the numerical diffusion speed near the discontinuities, although non-explicit, is only related to the bulk flow and plasma thermal speed. In very low-beta plasma simulations (e.g., $\beta < 10^{-4}$) such that magnetosonic waves dominate the dynamics, the use of the thermal speed in the gas-hydro distribution functions above is not adequate for describing the behavior of plasma associated with Alfvén waves. Thus, additional numerical diffusion is needed to damp spurious oscillations at the Alfvén speed when the left and right states are not equal. For gas-kinetic fluid flux functions, the numerical diffusion speed for the fluid flux is adjusted based on the average Alfvén speed at the interface, which is simply implemented via incorporating an additional diffusion term that follows a Rusanov-type numerical flux calculation centered at the cell interfaces:

$$\Delta U_F = \Delta U_F + \frac{1}{2} V_{A_{i+\frac{1}{2}}} \begin{pmatrix} \rho_L - \rho_R \\ \rho_L u_x^L - \rho_R u_x^R \\ \rho_L u_y^L - \rho_R u_y^R \\ \rho_L u_z^L - \rho_R u_z^R \\ E_P^L - E_P^R \end{pmatrix} \cdot \Delta t A_s, \quad (62)$$

where $V_{A_{i+\frac{1}{2}}}$ is chosen to be the average Alfvén speed V_A at the cell interface $i + \frac{1}{2}$:

$$V_{A_{i+\frac{1}{2}}} = \frac{1}{2} (V_A^L + V_A^R). \quad (63)$$

Note that the change of the momentum flux vector $\Delta U_F|_{\rho\mathbf{u}}$ in Equation (62) has already been transformed back to the base (x , y , z) system. Thus, the diffusion terms for the momentum fluxes are computed using the Cartesian components of the interface bulk velocities (\mathbf{u}^L and \mathbf{u}^R) directly. The additional diffusion terms in Equation (62) only apply to the interfaces where the left- and right-state variables are not equal. In smooth flow regions, the left- and right-state variables are equal or the differences are negligibly small, so no numerical diffusion is introduced to the interface fluxes through the above flux function.

3.4.2. Magnetic Stresses

The calculations of magnetic stress terms go through a similar process as computing the fluid fluxes, using similar Maxwellian-based, magneto-gas-kinetic distribution functions for the magnetic fields on the left and right sides of the interfaces (Xu 1999):

$$f_B^L(v) = \sqrt{\frac{\rho^L}{2\pi P_{\text{tot}}^L}} e^{-\frac{\rho^L}{2P_{\text{tot}}^L}(v - u_{x_n}^L)^2}, \quad (64)$$

$$f_B^R(v) = \sqrt{\frac{\rho^R}{2\pi P_{\text{tot}}^R}} e^{-\frac{\rho^R}{2P_{\text{tot}}^R}(v - u_{x_n}^R)^2}, \quad (65)$$

where P_{tot}^L and P_{tot}^R correspond to the left and right total pressure defined as $P_{\text{tot}}^L = \frac{1}{2}(B_x^{L2} + B_y^{L2} + B_z^{L2}) + P^L$ and $P_{\text{tot}}^R = \frac{1}{2}(B_x^{R2} + B_y^{R2} + B_z^{R2}) + P^R$, where (B_x^L, B_y^L, B_z^L) and (B_x^R, B_y^R, B_z^R) are the interface magnetic fields reconstructed from the cell-centered magnetic fields \mathbf{B}^{xyz} . This choice for the magnetic distribution function is arbitrary since the actual magnetic field is not a physically distributed quantity in the microscopic plasma velocity space v . The idea of introducing a velocity distribution function to the magnetic field is to spread the plasma information in the magnetic field such that the bulk plasma and the magnetic fields are coupled through the Alfvén speed.

In the magnetic flux functions, the magnetic stress tensor applied at the interface is calculated through similar moment integrals as in Equation (49) using the magneto-gas-kinetic distribution functions $f_B^L(v)$ and $f_B^R(v)$:

$$\begin{aligned} \bar{S}_{\text{mag}} = & \int_0^{+\infty} \left[\frac{1}{2} (B^L)^2 \bar{\mathbf{I}} - \mathbf{B}^L \mathbf{B}^L \right] f_B^L(v) dv \\ & + \int_{-\infty}^0 \left[\frac{1}{2} (B^R)^2 \bar{\mathbf{I}} - \mathbf{B}^R \mathbf{B}^R \right] f_B^R(v) dv \\ = & \langle v_B^0 \rangle_+^L \left[\frac{1}{2} (B^L)^2 \bar{\mathbf{I}} - \mathbf{B}^L \mathbf{B}^L \right] \\ & + \langle v_B^0 \rangle_-^R \left[\frac{1}{2} (B^R)^2 \bar{\mathbf{I}} - \mathbf{B}^R \mathbf{B}^R \right], \end{aligned} \quad (66)$$

where \mathbf{B}^L and \mathbf{B}^R are the Cartesian components of interface magnetic fields on the left and right sides of the interface, respectively. $(B^L)^2/2 = \frac{1}{2}[(B_x^L)^2 + (B_y^L)^2 + (B_z^L)^2]$ is the magnetic energy on the left side of the interface, and $(B^R)^2/2 = \frac{1}{2}[(B_x^R)^2 + (B_y^R)^2 + (B_z^R)^2]$ is the magnetic energy on the right side of the interface. $\langle v_B^0 \rangle_+^L$ and $\langle v_B^0 \rangle_-^R$ are zeroth

moments of the left and right magneto-gas-kinetic distribution functions:

$$\begin{aligned}\langle v_B^{0L} \rangle_+ &= \int_0^{+\infty} v^0 f_B^L(v) dv = \frac{1}{2} \operatorname{erfc}(-u_{x_n}^L \sqrt{\lambda_B^L}), \\ \langle v_B^{0R} \rangle_- &= \int_{-\infty}^0 v^0 f_B^R(v) dv = \frac{1}{2} \operatorname{erfc}(-u_{x_n}^R \sqrt{\lambda_B^R}),\end{aligned}\quad (67)$$

with $\lambda_B^L = \rho^L/2P_{\text{tot}}^L$ and $\lambda_B^R = \rho^R/2P_{\text{tot}}^R$. The evaluation of the moment integrals in Equation (66) for the magnetic stress $\bar{\mathbf{S}}_{\text{mag}}$ is straightforward since the magnetic stress tensor $(\frac{1}{2}B^2\bar{\mathbf{I}} - \mathbf{B}\mathbf{B})$ is not a function of the bulk velocity. Thus, the left and right magnetic stresses applied on the cell interfaces are calculated only using the zeroth moment of the magneto-gas-kinetic distribution functions $f_B^L(v)$ and $f_B^R(v)$.

Unlike the momentum flux calculations, Equation (66) for the magnetic stress $\bar{\mathbf{S}}_{\text{mag}}$ is evaluated directly in the base (x, y, z) coordinate system, because only the zeroth moment of the magneto-gas-kinetic distribution functions is involved. Thus, no face-normal coordinate transforms or inverse transforms are needed, as were used in the calculations of the fluid stress terms. According to Gauss's law, the volume-integrated Lorentz force $\Delta \mathbf{U}_B$ within a controlled cell V is the surface integral of the magnetic stress tensor

$$\begin{aligned}\Delta \mathbf{U}_B &= \int_V \nabla \cdot \left(\frac{1}{2} B^2 \bar{\mathbf{I}} - \mathbf{B}\mathbf{B} \right) dV \\ &= \oint \bar{\mathbf{S}}_{\text{mag}} \cdot d\mathbf{S} = \sum_{s=\mu, \nu, \zeta} (\bar{\mathbf{S}}_{\text{mag}} \cdot \mathbf{n}_s A_s) \\ &= \sum_{s=\mu, \nu, \zeta} \left[\langle v_B^{0L} \rangle_+ \left[\frac{1}{2} (B^L)^2 \bar{\mathbf{I}} - \mathbf{B}^L \mathbf{B}^L \right] \cdot \mathbf{n}_s A_s \right. \\ &\quad \left. + \langle v_B^{0R} \rangle_- \left[\frac{1}{2} (B^R)^2 \bar{\mathbf{I}} - \mathbf{B}^R \mathbf{B}^R \right] \cdot \mathbf{n}_s A_s \right] \\ &= \sum_{s=\mu, \nu, \zeta} \left(\langle v_B^{0L} \rangle_+ \left[\frac{1}{2} (B^L)^2 \mathbf{n}_s A_s - \mathbf{B}^L \Phi^s \right] \right. \\ &\quad \left. + \langle v_B^{0R} \rangle_- \left[\frac{1}{2} (B^R)^2 \mathbf{n}_s A_s - \mathbf{B}^R \Phi^s \right] \right).\end{aligned}\quad (68)$$

In the above flux function, $\mathbf{B}^L \cdot \mathbf{n}_s = \mathbf{B}^R \cdot \mathbf{n}_s = \Phi^s$ is the interface magnetic flux tracked in the Maxwell solver, which does not require the reconstruction and splitting procedure. The fact that $\mathbf{B}^L \cdot \mathbf{n}_s = \mathbf{B}^R \cdot \mathbf{n}_s$ is consistent with the requirement of the continuity condition of B_{x_n} in the one-dimensional flux function as discussed in Xu (1999).

The above calculations for the magnetic stress terms can be adapted to incorporate background magnetic field terms for very low- β plasma simulations, such as the inner magnetosphere of the Earth. The details of implementing background fields can be problem-specific but in general, a background field \mathbf{B}_0 can be included in the magnetic stress calculations using the following algorithm:

$$\begin{aligned}\Delta \mathbf{U}_B &= \int_V \nabla \cdot \left[\frac{1}{2} (\mathbf{B} + \mathbf{B}_0)^2 \bar{\mathbf{I}} - (\mathbf{B} + \mathbf{B}_0)(\mathbf{B} + \mathbf{B}_0) \right] dV \\ &= \oint \bar{\mathbf{S}}_{\text{mag}} \cdot d\mathbf{S}\end{aligned}\quad (69)$$

$$= \sum_{s=\mu, \nu, \zeta} (\bar{\mathbf{S}}_{\text{mag}} \cdot \mathbf{n}_s A_s) \quad (70)$$

$$\begin{aligned}&= \sum_{s=\mu, \nu, \zeta} \left(\langle v_B^{0L} \rangle_+ \left[\frac{1}{2} (B^{L2} + 2\mathbf{B}^L \cdot \mathbf{B}_0 + B_0^2) \mathbf{n}_s A_s \right. \right. \\ &\quad \left. \left. - (\mathbf{B}^L + \mathbf{B}_0)(\Phi^s + \Phi_{0s}) \right] \right) \\ &\quad + \sum_{s=\mu, \nu, \zeta} \left(\langle v_B^{0R} \rangle_- \left[\frac{1}{2} (B^{R2} + 2\mathbf{B}^R \cdot \mathbf{B}_0 + B_0^2) \mathbf{n}_s A_s \right. \right. \\ &\quad \left. \left. - (\mathbf{B}^R + \mathbf{B}_0)(\Phi^s + \Phi_{0s}) \right] \right),\end{aligned}\quad (71)$$

where $\frac{1}{2}B_0^2 = (B_{0x}^2 + B_{0y}^2 + B_{0z}^2)/2$ and $\Phi_{0s} = \mathbf{B}_0 \cdot \mathbf{n}_s$ are the magnetic energy and the magnetic flux of the background field at interface s , respectively. The background magnetic field components \mathbf{B}_0 on the cell interfaces used in Equation (71) are computed using a 12th-order 2D Gaussian quadrature rather than point evaluations, to improve the implementation of force-free background magnetic fields, i.e., $\nabla \times \mathbf{B}_0 = 0$. For example, at μ -faces,

$$\mathbf{B}_{0_{i+\frac{1}{2},j,k}}^\mu = \frac{1}{A_\mu} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \mathbf{B}_0 \left(\mu = i + \frac{1}{2}, \nu, \zeta \right) d\nu d\zeta, \quad (72)$$

and the magnetic energy of the background at the corresponding cell interfaces are computed as

$$(B_{0_{i+\frac{1}{2},j,k}}^\mu)^2 = \frac{1}{A_\mu} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} B_0^2 \left(\mu = i + \frac{1}{2}, \nu, \zeta \right) d\nu d\zeta, \quad (73)$$

and the cross terms of the background magnetic fields in Equation (71) are calculated as

$$\begin{aligned}\mathbf{B}_{0_{i+\frac{1}{2},j,k}}^\mu \mathbf{B}_{0_{i+\frac{1}{2},j,k}}^\mu &= \frac{1}{A_\mu} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \\ &\quad \times \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \mathbf{B}_0 \left(\mu = i + \frac{1}{2}, \nu, \zeta \right) \mathbf{B}_0 \left(\mu = i + \frac{1}{2}, \nu, \zeta \right) d\nu d\zeta.\end{aligned}\quad (74)$$

3.4.3. Face-normal Coordinate Systems

The MHD solver in the GAMERA code only tracks Cartesian components of the plasma velocity (u_x, u_y, u_z) regardless of the grid geometry. However, as discussed in the previous section, the calculations of fluid flux are done in the local face-normal coordinate system (x_n, x_1, x_2) , which is usually not the same as the base Cartesian system. Thus, after the interface reconstruction of the Cartesian velocity components (u_x, u_y, u_z) and magnetic field (B_x, B_y, B_z) in the μ -, ν - and ζ -directions, corresponding coordinate transforms at the cell interfaces are needed to rotate the left and right velocity and magnetic field from the base Cartesian coordinate system (x, y, z) into the face-normal coordinate system (x_n, x_1, x_2) , in order to evaluate the numerical fluxes using the gas-kinetic flux functions described in Section 3.4. For updating the plasma momentum after computing the momentum flux vector in the local face-normal coordinate systems at each interface, inverse transforms are needed to rotate the face-integrated momentum flux from the (x_n, x_1, x_2) system to the base Cartesian system (x, y, z) . Then, the Cartesian components of the plasma momentum $(\rho u_x, \rho u_y, \rho u_z)$ are updated as described in the previous section. This transform-inverse transform algorithm enables the MHD solver to evolve the plasma velocity components in the base

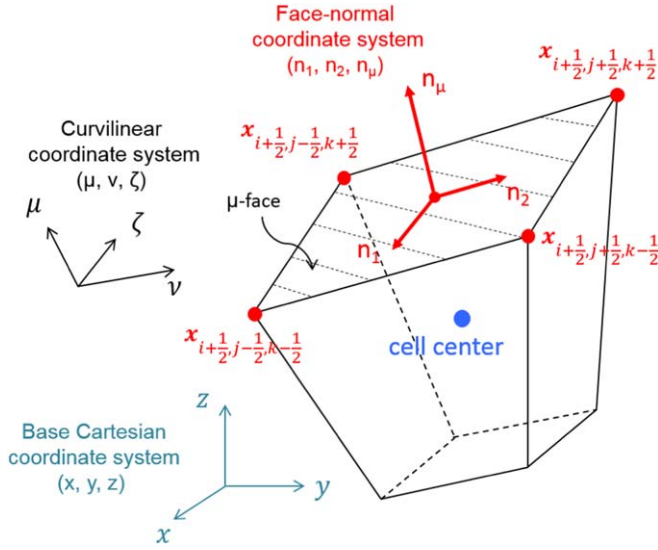


Figure 7. The local μ -face-normal coordinate system ($\mathbf{n}_\mu, \mathbf{n}_1, \mathbf{n}_2$) in a finite-volume cell indexed as (i, j, k) .

Cartesian coordinate system, which is independent of the curvilinear coordinate system used to define the computational grid. Therefore, orthogonality of the curvilinear grid is not required in the MHD solver, and the numerical grid can be adapted to simulate problem-specific flow patterns, such as planetary magnetospheres.

Using the orthogonal μ -face-normal coordinate system ($\mathbf{n}_\mu, \mathbf{n}_1, \mathbf{n}_2$) at cell interfaces indexed by $(i + \frac{1}{2}, j, k)$ is illustrated in Figure 7 as an example, where \mathbf{n}_μ is the unit vector normal to the μ -face, and \mathbf{n}_1 and \mathbf{n}_2 are the two unit vectors perpendicular to \mathbf{n}_μ . The $\mathbf{n}_1, \mathbf{n}_2$, and \mathbf{n}_μ vectors from the orthogonal system, which are calculated using the four corner grid points forming the μ -face shown in Figure 7, are as follows:

$$\mathbf{n}_\mu = \frac{(\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k-\frac{1}{2}}) \times (\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}})}{|\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k-\frac{1}{2}}| \times |\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}}|} \quad (75)$$

$$\mathbf{n}_2 = \frac{\mathbf{n}_\mu \times (\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}})}{|\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}}|} \quad (76)$$

$$\mathbf{n}_1 = \mathbf{n}_2 \times \mathbf{n}_\mu. \quad (77)$$

Since $||\mathbf{n}_1|| = ||\mathbf{n}_2|| = ||\mathbf{n}_\mu|| \equiv 1$ and $\mathbf{n}_1 \perp \mathbf{n}_2 \perp \mathbf{n}_\mu$, the transform of the interface velocity vectors from the base Cartesian (x, y, z) coordinate system to the face-normal $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_\mu)$ coordinate system is done as follows:

$$\begin{aligned} \mathbf{u}_{n_1, n_2, n_\mu} &= \bar{\mathbf{T}} \cdot \mathbf{u}_{x, y, z} \Rightarrow \begin{pmatrix} u_{n_1} \\ u_{n_2} \\ u_{n_\mu} \end{pmatrix} \\ &= \begin{bmatrix} n_1^x & n_1^y & n_1^z \\ n_2^x & n_2^y & n_2^z \\ n_\mu^x & n_\mu^y & n_\mu^z \end{bmatrix} \cdot \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \end{aligned} \quad (78)$$

where $\mathbf{u}_{n_1, n_2, n_\mu} = (u_{n_1}, u_{n_2}, u_{n_\mu})$ is the velocity vector in the face-normal coordinate system $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_\mu)$ at the μ -faces, and

$\mathbf{u}_{x, y, z} = (u_x, u_y, u_z)$ is the corresponding velocity vector in the base Cartesian coordinate system (x, y, z) . Since the transform matrix $\bar{\mathbf{T}}$ in the above equation is an orthogonal matrix, the inverse transform to rotate the stress $\mathbf{F}_{\text{FLUID}}$ from the $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_\mu)$ system to the base (x, y, z) system is simply done using the transpose of $\bar{\mathbf{T}}$:

$$\begin{aligned} \mathbf{F}_{\rho u}^{x, y, z} &= \bar{\mathbf{T}}^T \cdot \mathbf{F}_{\rho u}^{n_1, n_2, n_\mu} \Rightarrow \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} \\ &= \begin{bmatrix} n_1^x & n_2^x & n_\mu^x \\ n_1^y & n_2^y & n_\mu^y \\ n_1^z & n_2^z & n_\mu^z \end{bmatrix} \cdot \begin{pmatrix} F_{n_1} \\ F_{n_2} \\ F_{n_\mu} \end{pmatrix}, \end{aligned} \quad (79)$$

where $\mathbf{F}_{\rho u}^{n_1, n_2, n_\mu}$ is the momentum flux vector in the $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_\mu)$ coordinate system described in Equation (60), and $\mathbf{F}_{\rho u}^{x, y, z}$ is the corresponding vector of momentum flux in the base Cartesian coordinate system (x, y, z) . With the coordinate transform, the surface integral of the fluid stress for $\Delta U_F|_{\rho u}$ is simply calculated using (61). However, the surface integrals of mass flux and energy flux are in the \mathbf{n}_μ direction, and they are used to compute the surface integrals directly without coordinate transforms, as shown in Equations (58) and (59). For the magnetic stresses, since only the zeroth moment of the distribution function is involved, the stress calculations on the cell interfaces are done directly using the x -, y -, and z -components of the interface magnetic fields and the $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_\mu$ vectors in the base Cartesian system as shown in Equation (68), without the above transform/reverse transform procedures.

3.5. Constrained Transport for Non-orthogonal Grids

In this section, we describe the numerical schemes for calculating the electric fields defined at the cell edges for

evolving the magnetic flux Φ through Faraday's law. The calculation of the electric field is handled through the implementation of a constrained transport method based on the same high-order reconstruction schemes (described in Section 3.3.1) adapted to non-orthogonal staggered grids. As a main part of the corrector step illustrated in Figure 2, the predicted velocity $\mathbf{u}^{i+\frac{1}{2}}$ and magnetic flux $\Phi^{i+\frac{1}{2}}$ are used in the Maxwell solver for computing the electric fields. Figure 8 shows the algorithms of computing the electric field defined at the cell edges using the predicted bulk velocity $\mathbf{u}^{n+\frac{1}{2}}$ and magnetic flux $\Phi^{n+\frac{1}{2}}$, with details described in the following sections.

3.5.1. Calculation of Electric Fields

Figure 9 shows a schematic of the grid geometry for computing the electric field component E_ζ at ζ -edges. The E_μ and E_ν components along the μ - and ν -edges are calculated using the same algorithm as E_ζ , with corresponding rotations of the computational grid. Using the example shown in Figure 9, the electric field E_ζ is located at the cell edge indexed as

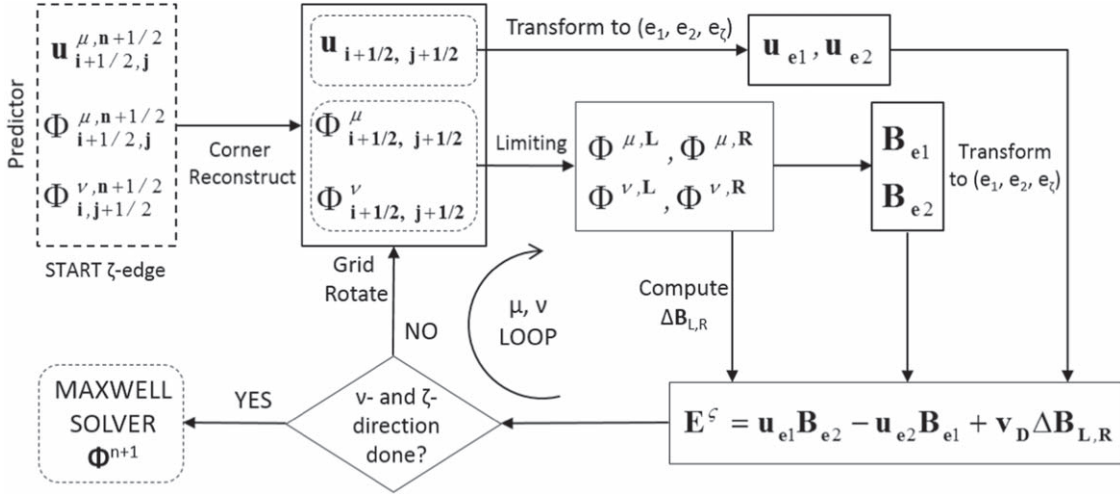


Figure 8. The algorithm for calculating the edge-aligned electric field using the predictors.

$(i + \frac{1}{2}, j + \frac{1}{2}, k)$, which is surrounded by four neighboring control volumes whose cell centers are indexed as (i, j, k) , $(i + 1, j, k)$, $(i, j + 1, k)$, and $(i + 1, j + 1, k)$, respectively. To obtain a high-order estimation of the electric field component E_ζ at the ζ -edge, both face-centered magnetic flux and cell-centered plasma velocity are reconstructed at the cell edges using the high-order reconstruction method described in Section 3.3.1. For the velocity $\mathbf{u}(i + \frac{1}{2}, j + \frac{1}{2}, k)$ at the cell edges, this is done by first performing a high-order reconstruction to the μ -interfaces and then reconstructing these μ -interface values in the ν -direction toward the edge (or the corner as it appears in the top view of the 2D slice shown in the right panel of Figure 9):

$$\rho \mathbf{u}(i, j, k) \xrightarrow{\text{interp in } \mu} \rho \mathbf{u}\left(i + \frac{1}{2}, j, k\right) \xrightarrow{\text{interp in } \nu} \rho \mathbf{u}\left(i + \frac{1}{2}, j + \frac{1}{2}, k\right) \xrightarrow{\text{divide by } \rho} \mathbf{u}\left(i + \frac{1}{2}, j + \frac{1}{2}, k\right). \quad (80)$$

The default choice for reconstructing the edge velocity is an eighth-order centered scheme described in Equation (37). The estimation of the magnetic field vectors at the cell edge is more complicated than computing the velocity vectors, which requires two reconstruction steps for both magnetic flux and face-normal vectors. The first step is to reconstruct the magnetic flux at the cell edges. Since the magnetic flux Φ^μ and Φ^ν are already defined on the cell interfaces, they only require a single reconstruction to the corner along the ν - and μ -directions, respectively. Using the Φ^μ component of the magnetic flux along the ν -direction, as shown in Figure 10, after applying the one-dimensional reconstruction algorithm to the magnetic flux $\Phi_{i+\frac{1}{2},j,k}^\mu$ in the ν -direction, two “interface” states of the magnetic flux, $\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,L)}$ and $\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,R)}$ are computed at the cell edge $(i + \frac{1}{2}, j + \frac{1}{2}, k)$. To estimate the magnetic field strength at cell edges, an eighth-order interpolation is applied to the face area $A_{i+\frac{1}{2},j,k}^\mu$ along the ν -direction to find an estimation for the area $A_{i+\frac{1}{2},j+\frac{1}{2},k}^\mu$ at the cell edge for the estimated interface fluxes $\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,L)}$ and $\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,R)}$, as shown by the shaded area in Figure 10. Then, the magnetic field

strength in the μ -direction at the cell edge $(i + \frac{1}{2}, j + \frac{1}{2}, k)$ is calculated as the average of the left and right state of the magnetic flux divided by the estimated interface area:

$$B_{\text{avg}}^\mu = \frac{1}{2} \frac{\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,L)} + \Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\mu,R)}}{A_{i+\frac{1}{2},j+\frac{1}{2},k}^\mu}. \quad (81)$$

The next step is to estimate the direction of B_{avg}^μ at the cell edge $(i + \frac{1}{2}, j + \frac{1}{2}, k)$, which is denoted as $\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}$ in Figure 10. This is done by interpolating the face-normal unit vectors $\hat{\mathbf{n}}_\mu$ to the cell edge using the same eighth-order reconstruction algorithm. The expression for the $\hat{\mathbf{n}}_\mu$ vectors is defined in Equation (75), and the high-order reconstructed $\hat{\mathbf{n}}_\mu$ vector at the cell edge is denoted as $\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}$. Similarly, the average magnetic field in the ν -direction is calculated by applying the same reconstruction algorithm on the magnetic flux component $\Phi_{i,j+\frac{1}{2},k}^\nu$ and the corresponding ν -face area $A_{i,j+\frac{1}{2},k}^\nu$ the μ -direction:

$$B_{\text{avg}}^\nu = \frac{1}{2} \frac{\Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\nu,L)} + \Phi_{i+\frac{1}{2},j+\frac{1}{2},k}^{(\nu,R)}}{A_{i+\frac{1}{2},j+\frac{1}{2},k}^\nu}. \quad (82)$$

The direction of B_{avg}^ν is also interpolated from the face-normal $\hat{\mathbf{n}}_\nu$ vectors using the same high-order interpolation schemes. The high-order interpolated $\hat{\mathbf{n}}_\nu$ vector at the cell edge is denoted as $\hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}$.

Using the reconstructed edge velocity and average magnetic field vector, the calculation of the ζ -component of the electric field is basically a Rusanov scheme adapted to a cell corner. With the edge velocity vector $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ and mean magnetic fields $B_{\text{avg}}^{\mu,\nu}$, the electric field component E_ζ at the ζ -edge is calculated as

$$E_\zeta = -\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k} \times \mathbf{B}_{\text{avg}}^{\mu,\nu} + \eta_A \mathbf{j}_\zeta, \quad (83)$$

where \mathbf{j}_ζ is the component of $\nabla \times \mathbf{B}$ along the ζ -edge computed using the left and right states of the μ - and ζ -edge magnetic field as shown in the right panel of Figure 9, and η_A is a numerical resistivity dealing with the propagation of Alfvén waves near discontinuities. If computing the edge electric field using Equation (83) without the “resistive” term, simulations

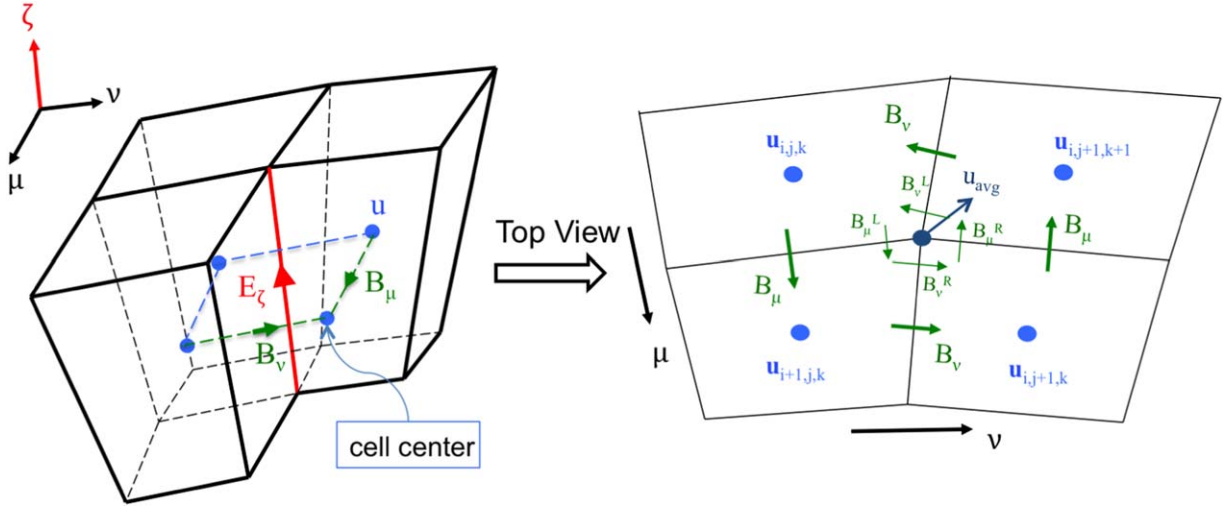


Figure 9. A schematic showing the locations of the various quantities needed for the calculation of the electric field at the cell edges.

tend to develop spurious oscillations in the MHD flows where Alfvén waves are dominant, especially when discontinuities occur. We should note that this “resistive” term $\eta_A \mathbf{j}$ in Equation (83) only operates when the limiter detects a discontinuous situation. When the flow is smooth, $\mathbf{B}_\mu^L = \mathbf{B}_\mu^R$ and $\mathbf{B}_\nu^L = \mathbf{B}_\nu^R$, the \mathbf{j}_ζ component in the diffusion term is essentially zero. Therefore, in smooth regions the electric field, E_ζ is just a high-order approximation of $-\mathbf{u} \times \mathbf{B}$ computed at the cell edges.

In Equation (83), the interpolated velocity $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ and the average magnetic field \mathbf{B}_{avg} at the ζ -edge are not in the same coordinate system; the velocity $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ is in the base Cartesian coordinate system (x, y, z) , while the magnetic field \mathbf{B}_{avg} is in a local non-orthogonal coordinate system $(\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}, \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}, \zeta)$, as illustrated by the green arrows in Figure 11. Therefore, to calculate the cross product between the velocity and magnetic field at the cell edges, we define a new orthogonal coordinate system $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$ with one axis \mathbf{e}_ζ aligned with the direction of the cell edge, and we transform $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ and $\mathbf{B}_{\text{avg}}^{\mu,\nu}$ into this new coordinate system. Using the ζ -edge-aligned orthogonal system shown in Figure 11 as an example, the local coordinate system $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$ for computing the ζ -component of the electric field using Equation (83) is shown in red, where E_ζ is along the direction of \mathbf{e}_ζ . For the ζ -edge indexed by $(i + \frac{1}{2}, j + \frac{1}{2}, k)$, the $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$ coordinate system is computed as follows. First, compute the ζ -edge unit vector \mathbf{e}_ζ :

$$\mathbf{e}_\zeta = \frac{\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}}{|\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}|}. \quad (84)$$

Then, calculate the direction \mathbf{e}_1 normal to the ζ vector

$$\mathbf{e}_1 = \frac{(\mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k}^E - \mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k}^E) \times \mathbf{e}_\zeta}{|\mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k}^E - \mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k}^E|}, \quad (85)$$

where $\mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k}^E$ and $\mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k}^E$ are the edge-centered positions \mathbf{A} and \mathbf{B} at two neighboring ζ -edges shown in Figure 11:

$$\mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k}^E = \frac{1}{2}(\mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}}), \quad (86)$$

$$\mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k}^E = \frac{1}{2}(\mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k+\frac{1}{2}} + \mathbf{x}_{i+\frac{1}{2},j+\frac{3}{2},k-\frac{1}{2}}). \quad (87)$$

Then, the third direction \mathbf{e}_2 , orthogonal to both \mathbf{e}_1 and \mathbf{e}_ζ , is obtained using the following cross product:

$$\mathbf{e}_2 = \mathbf{e}_\zeta \times \mathbf{e}_1. \quad (88)$$

Since the reconstructed edge velocity $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ is in the base Cartesian system, it is straightforward to map the reconstructed edge velocity vector $\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k}$ into the new edge-aligned coordinate system $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$:

$$u_{e1} = \mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k} \cdot \mathbf{e}_1, \quad (89)$$

$$u_{e2} = \mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k} \cdot \mathbf{e}_2, \quad (90)$$

where u_{e1} and u_{e2} are the velocity components in the directions of \mathbf{e}_1 and \mathbf{e}_2 , respectively. The $u_{e\zeta}$ component is along the ζ edge, which does not contribute to the calculation of the E_ζ component. The edge magnetic field components $\mathbf{B}_{\text{avg}}^\mu$ and $\mathbf{B}_{\text{avg}}^\nu$ calculated using Equations (81) and (82) are in a non-orthogonal local coordinate system $(\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}, \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}, \zeta)$; thus, the coordinate transform is more complicated than the velocity components, which require solving the following 2×2 system:

$$\mathbf{B}_{\text{avg}}^\mu = \xi_1^\mu \mathbf{B}_{\text{avg}}^{e1} + \xi_2^\mu \mathbf{B}_{\text{avg}}^{e2}, \quad (91)$$

$$\mathbf{B}_{\text{avg}}^\nu = \xi_1^\nu \mathbf{B}_{\text{avg}}^{e1} + \xi_2^\nu \mathbf{B}_{\text{avg}}^{e2}, \quad (92)$$

where ξ^μ and ξ^ν are the projections of the directions of the magnetic field $\mathbf{B}_{\text{avg}}^{\mu,\nu}$ in the $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$ coordinate system:

$$\xi_1^\mu = \hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \cdot \mathbf{e}_1 \quad \xi_2^\mu = \hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \cdot \mathbf{e}_2, \quad (93)$$

$$\xi_1^\nu = \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \cdot \mathbf{e}_1 \quad \xi_2^\nu = \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \cdot \mathbf{e}_2. \quad (94)$$

After obtaining the average magnetic field components $\mathbf{B}_{\text{avg}}^{e1}$ and $\mathbf{B}_{\text{avg}}^{e2}$ in the new edge-aligned coordinate system $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_\zeta)$, the electric field E_ζ along the edge \mathbf{e}_ζ is calculated based on

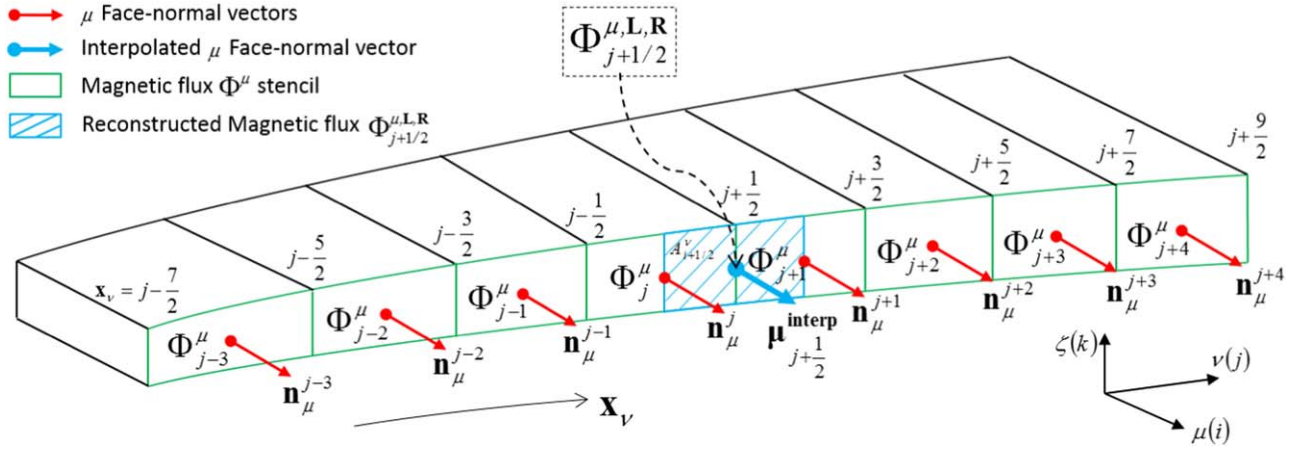


Figure 10. The one-dimensional reconstruction stencil for Φ^μ and \mathbf{n}_μ in the ν -direction for estimating $\mathbf{B}_{\text{avg}}^\mu$.

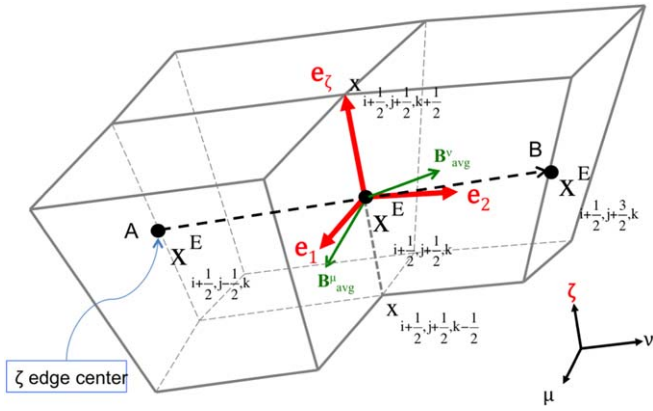


Figure 11. The edge-aligned coordinate system (\mathbf{e}_1 , \mathbf{e}_2 , \mathbf{e}_ζ) for computing the electric field.

Equation (83):

$$E_\zeta = -(u_{e1} B_{\text{avg}}^{e2} - u_{e2} B_{\text{avg}}^{e1}) + v_D (B_R^\mu - B_L^\mu + B_L^\nu - B_R^\nu). \quad (95)$$

Then, the electric potential along the ζ edge is calculated as

$$\mathcal{E}_{i+\frac{1}{2},j+\frac{1}{2},k}^\zeta = E_\zeta |\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}|, \quad (96)$$

which is the edge-centered electric field E_ζ multiplied by the length of the corresponding ζ edge. $\mathcal{E}_{i+\frac{1}{2},j+\frac{1}{2},k}^\zeta$ is the actual component used in the integral form of Faraday's law to evolve the face-centered magnetic fluxes Φ . The diffusive speed v_D in Equation (95) is defined as

$$v_D = \frac{1}{2} (V_A + |\mathbf{u}|) \frac{|\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \times \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}|}{|\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}| |\hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}|}. \quad (97)$$

The magnitude of the diffusive speed is usually set to reflect the magnitude of the local convection speed and Alfvén speed. The $|\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}} \times \hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}| / |\hat{\mu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}| |\hat{\nu}_{i+\frac{1}{2},j+\frac{1}{2},k}^{\text{interp}}|$ term originates from the $\nabla \times$ operation in the coordinate transform of the non-orthogonal geometry when the diffusive current \mathbf{j}_ζ in Equation (83) is computed.

3.5.2. Evolution of Magnetic fluxes

The corresponding ν -edge- and μ -edge-aligned orthogonal coordinate systems are calculated in a similar manner as for the ζ edge. The electric potentials \mathcal{E}^μ and \mathcal{E}^ν along the μ and ν edges are also computed in the same way using the edge-aligned coordinate system. These electric potential are calculated using the same subroutine as for \mathcal{E}^ζ through proper rotation of the computational grid. Once all of the electric fields are calculated, the magnetic flux threading a face can be updated via Faraday's law:

$$\begin{aligned} \Phi_{i,j,k+\frac{1}{2}}^{\zeta,n+1} &= \Phi_{i,j,k+\frac{1}{2}}^{\zeta,n} + \Delta t (\mathcal{E}_{i,j-\frac{1}{2},k+\frac{1}{2}}^\nu + \mathcal{E}_{i+\frac{1}{2},j,k+\frac{1}{2}}^\mu \\ &\quad - \mathcal{E}_{i,j+\frac{1}{2},k+\frac{1}{2}}^\nu - \mathcal{E}_{i-\frac{1}{2},j,k+\frac{1}{2}}^\mu), \\ \Phi_{i+\frac{1}{2},j,k}^{\mu,n+1} &= \Phi_{i+\frac{1}{2},j,k}^{\mu,n} + \Delta t (\mathcal{E}_{i+\frac{1}{2},j,k-\frac{1}{2}}^\zeta + \mathcal{E}_{i+\frac{1}{2},j+\frac{1}{2},k}^\nu \\ &\quad - \mathcal{E}_{i+\frac{1}{2},j,k+\frac{1}{2}}^\zeta - \mathcal{E}_{i+\frac{1}{2},j-\frac{1}{2},k}^\nu), \\ \Phi_{i,j+\frac{1}{2},k}^{\nu,n+1} &= \Phi_{i,j+\frac{1}{2},k}^{\nu,n} + \Delta t (\mathcal{E}_{i-\frac{1}{2},j+\frac{1}{2},k}^\mu + \mathcal{E}_{i,j+\frac{1}{2},k+\frac{1}{2}}^\zeta \\ &\quad - \mathcal{E}_{i+\frac{1}{2},j+\frac{1}{2},k}^\mu - \mathcal{E}_{i,j+\frac{1}{2},k-\frac{1}{2}}^\zeta). \end{aligned} \quad (98)$$

Based on Equation (98), it is straightforward to show that the divergence of the magnetic field defined by the flux follows

$$\nabla \cdot \mathbf{B} |_{i,j,k}^{n+1} = \nabla \cdot \mathbf{B} |_{i,j,k}^n, \quad (99)$$

ensuring that the volume-integrated $\nabla \cdot \mathbf{B}$ is unmodified during the magnetic field evolution step, regardless of changes in the local electric fields. Therefore, as long as the initial divergence of the magnetic field is zero, the Maxwell solver keeps the $\nabla \cdot \mathbf{B}$ term to the round-off error automatically.

3.6. Implementation Considerations

Undertaking the task of completely rebuilding our simulation framework, while costly, presents significant opportunities. We have used this opportunity to modernize the software design, improve computational performance, and incorporate numerous algorithmic advances. Redesigning our code to prepare for the multicore era of supercomputing while maintaining a flexible and extensible code base has been a primary focus of this effort. The production implementation of GAMERA has been rewritten entirely from scratch in modern (2003/2008 standard) Fortran. Utilities that make up the pre- and post-

processing ecosystem are implemented in *Python*, with “heavy” data stored in the HDF5 format and interpreted via XDMF. The computational considerations necessary for multicore performance can often increase code complexity. We alleviate these difficulties by incorporating into our development the use of unit testing using the pFUnit framework (Rilee & Clune 2014).

The core computational routines and data structures of GAMERA have been designed to expose the multiple layers of the heterogeneous parallelism necessary for modern multicore architectures. At the highest level, spatial domain and model decomposition are undertaken via traditional MPI parallelism. However, in a multicore framework, it is inappropriate to use this higher-level parallelism across the light-weight compute cores within a socket. Instead, we utilize shared memory parallelism, OpenMP, to create finer-grained parallelism within the individual computations done on the compute domain of an MPI rank. Typically, we utilize 1 MPI rank per compute socket, and 1 thread per *virtual* core. Finally, and often most importantly, is the necessity of vector or SIMD (single-instruction multiple-data) parallelism. Modern computational architecture is optimized to move through memory in a predictable, ideally linear, manner and perform the same computations on each memory location. Code written to take advantage of this can easily outperform naive implementations by an order of magnitude.

Two major design choices allow GAMERA to fully take advantage of vector parallelism. The first is HPC-friendly data structures, namely the use of large, contiguous arrays. The main data structure within GAMERA is a 5D array of size $N_s \times N_v \times N_k \times N_j \times N_i$, where the dimensions represent the number of ion species, the number of plasma flow variables, and the spatial dimensions of the domain, respectively. The second choice is the manner in which our “heavy” compute routines are written, e.g., the interface flux calculation. These routines are written to work on memory-aligned blocks of data, typically twice the vector length of the architecture. This ensures that even very complex computational routines will be properly converted to vector instructions by the compiler while avoiding difficult-to-maintain manually inlined code.

The main computational workload of GAMERA, and typically most codes, is done in a series of loop nests. As an example, a sweep of the computational domain may take the ordering, from outer- to innermost, of (k, j, i) . In GAMERA, the innermost loop would be separated into i_B ranging over $[1, N_i/N_B]$, where N_i is the number of cells in the i -direction, N_B is the blocking size, and for δi , which ranges over $[1, N_B]$. This results in a loop ordering $(k, j, i_B, \delta i)$, with the δi loop within a subroutine called from the i_B loop. In all of our computationally intensive loop nests, the innermost loop is over δi regardless of the coordinate direction of the sweep, i.e., reconstruction in the j and k directions still utilize δi as the innermost loop, although the outermost loops may be reordered to improve cache locality. As an example, calculating fluxes in the j direction uses a loop nest of the form $(k, i_B, j, \delta i)$.

The result of these considerations is a code that significantly outperforms the original LFM and has been used to do production science simulations up to 10k CPU-cores on NCAR’s Cheyenne supercomputer. Performance optimization, however, is an iterative process and one largely undertaken upon completion of the core algorithm. The performance metrics we will discuss here are not intended to be definitive or

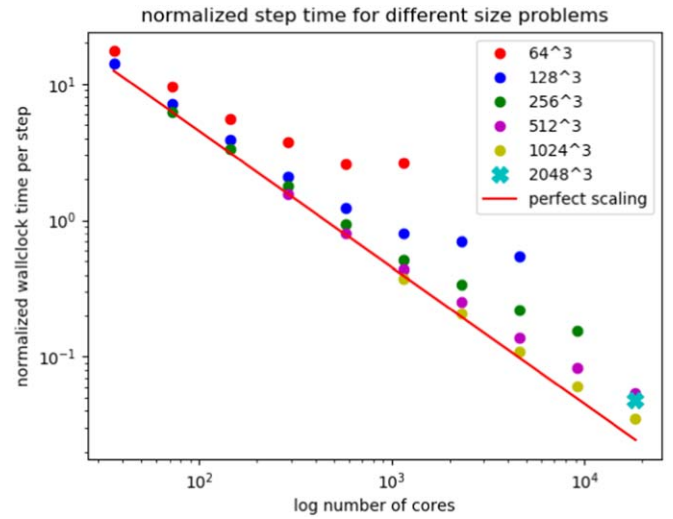


Figure 12. Strong-scaling for a variety of problem sizes.

final but merely a snapshot of the code’s performance upon the completion of main algorithm development. Even so, we find that the code scales well in both OMP and MPI and is capable of production scientific simulations at $O(10k)$ CPU-cores.

Our performance analysis has been undertaken on NCAR’s Cheyenne supercomputer, which utilizes Intel Xeon E5-2697V4 (Broadwell) processors, featuring dual-socket nodes with 18 CPU-cores per socket. Using one MPI rank/socket and two OMP threads per CPU core, we find OMP scaling at 85% of optimal and a performance of approximately 500k zone-cycles per second per core. The OpenMP scaling can likely be improved by merging thread-parallel regions to reduce fork/join overhead; however, this can come at the cost of code complexity and maintainability. Turning to cross-socket performance, Figure 12 shows strong-scaling for a collection of problem sizes using triply periodic MHD. This includes both computation and communication times, with the former scaling better individually. We note that we are not currently using OMP threading in our communication routines and expect that threaded packing/unpacking of buffers and utilization of thread-safe asynchronous MPI calls will further improve our scaling. Even so, we see near-linear scaling for all problem sizes until we cross the threshold of approximately 16^3 cells per compute core.

4. Test Results

In this section, we show one set of test simulations of two-dimensional linear advection and four sets of simulation results from two-dimensional standard MHD test problems in both Cartesian and non-Cartesian geometries, including field-loop advection, circularly polarized nonlinear Alfvén waves, the Orszag–Tang vortex, and spherical blast waves in strong magnetic fields. The 2D linear advection tests demonstrate the choice of high-order reconstruction methods discussed in Section 3.3.1. Both the field-loop advection and nonlinear Alfvén wave simulations provide quantitative assessment for the numerical solutions, while the Orszag–Tang and blast-wave simulations are less quantitative. The latter two sets of test simulations are used to demonstrate the effectiveness of the numerical schemes on handling highly nonlinear MHD flows in non-orthogonal distorted grid geometries.

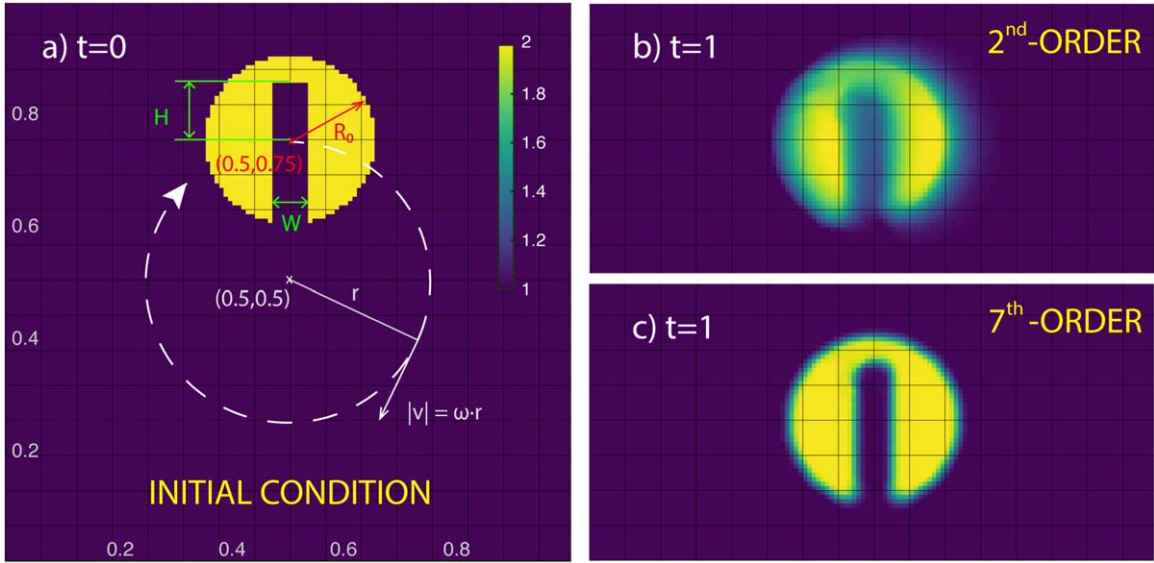


Figure 13. Panel (a) shows the initial distribution of a slotted cylinder in Cartesian geometry with 128×128 cells at $t = 0$; panel (b) shows the simulated density distribution at $t = 1$ using a second-order reconstruction scheme; and panel (c) shows the simulated density distribution at $t = 1$ using a seventh-order reconstruction scheme. Each “grid cell” shown in panels (a)–(c) using thin black lines contains 8×8 actual computational cells in the x and y directions, respectively.

4.1. Circular Advection

We use the circular advection of a slotted cylinder problem used by Colella et al. (2011) as a multidimensional linear test to show the effectiveness of the advection scheme, which was originally developed by Zalesak (1979). This test problem is also used to demonstrate the necessity of the high-order reconstruction method in the numerical algorithms. In this two-dimensional circular advection test, only the mass continuity equation is solved

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v}_0), \quad (100)$$

where \mathbf{v}_0 is a time-stationary circular velocity field defined as $\mathbf{v}_0 = -2\pi\omega\hat{\theta}$ with $\omega = 1$ the angular velocity and $r = \sqrt{(x - 0.5)^2 + (y - 0.5)^2}$ as shown in Figure 13(a). The simulation domain is $0 \leq x \leq 1$, $0 \leq y \leq 1$ with periodic boundary conditions in both the x and y directions. The initial density distribution at $t = 0$ is defined as a slotted cylinder of radius $R_0 = 0.15$ centered at $(x, y) = (0.5, 0.75)$, with slot width $W = 0.06$ and slot height $H = 0.1$:

$$\rho(x, y)|_{t=0} = \begin{cases} 1, & r \geq R_0, \\ 1, & |x - 0.5| \leq W \text{ and } y \leq 0.75 + H, \\ 2, & \text{otherwise.} \end{cases} \quad (101)$$

At $t = 1$, the center of the slotted cylinder returns to the initial location. The initial solution on a uniform Cartesian mesh with 128×128 cells is plotted in Figure 13(a). The solution of ρ (at $t = 1$) using the second-order centered reconstruction method introduced in Section 3.3.1 is shown in Figure 13(b), while the solution of ρ at $t = 1$ using the default seventh-order reconstruction scheme is shown in Figure 13(c). At $t = 1$, although the second-order reconstruction scheme preserves the basic shape of the slotted cylinder, the initially sharp edges of the cylinder are smeared significantly, with noticeable fill-in of the slot. When using the seventh-order reconstruction scheme,

the density distribution at $t = 1$ is preserved more accurately compared to the second-order case, with sharper edges of the cylinder and an empty slot.

Figure 14 shows more quantitative comparisons of the circular advection results using line profiles of the simulated ρ at $y = 0.75$. The blue profile shows the initial density distribution at $t = 0$ as a reference, while the green and red profiles show the corresponding density profiles at $t = 1$ using the second- and the seventh-order reconstruction scheme, respectively. The comparison shows that the second-order reconstruction scheme introduces a significant amount of numerical diffusion. At $t = 1$, the profile of the slotted density is smeared using the second-order reconstruction, which has approximately 14 cells resolving the contact discontinuity at the edge of the cylinder ($x = 0.5 \pm R_0$) and a filled slot with density ≈ 1.3 ($|x - 0.5| \leq R_0$). On the other hand, the seventh-order reconstruction scheme preserves the initial profile well, with only four cells resolving the sharp edge of the cylinder and an empty slot (density ≈ 1.0). The comparison of the density profiles between the second- and seventh-order reconstruction scheme demonstrates the necessity of using very high-order reconstruction methods in order to reduce the numerical diffusion and resolve sharp contact discontinuity in multi-dimensional flow simulations. The additional computing cost of the seventh-order reconstruction scheme is approximately 24% compared to the second-order scheme. When using the eighth-order reconstruction scheme, the additional computational cost is approximately 12%.

The initial and final solutions for the same two-dimensional circular advection simulation on three non-Cartesian grids are shown in Figure 15. Figures 15(a) and (d) are the results using a distorted Cartesian grid with 128×128 cells (Colella et al. 2011). Figures 15(b) and (e) use a mapped grid that deforms a Cartesian domain with 128×128 cells into a cylindrical domain (Calhoun et al. 2008). Figures 15(c) and (f) are from a spherical polar grid (this simulation uses 72×192 cells in order to obtain a spatial resolution of the slotted cylinder similar to those of the other two grids using 128×128 cells). This set of simulation results shows that the advection scheme

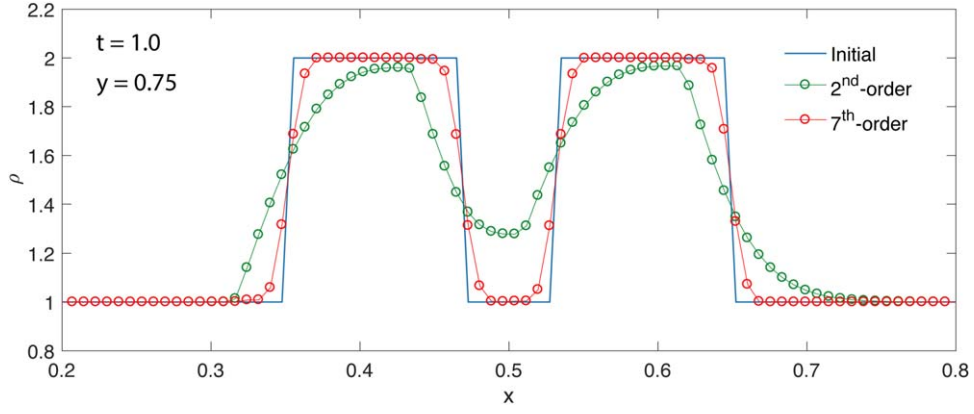


Figure 14. The comparison of density profiles at $y = 0.75$. The blue profile is the reference density distribution at $t = 0$, and the green and red profiles are the corresponding density distribution at $t = 1$ using the second- and seventh-order schemes, respectively.

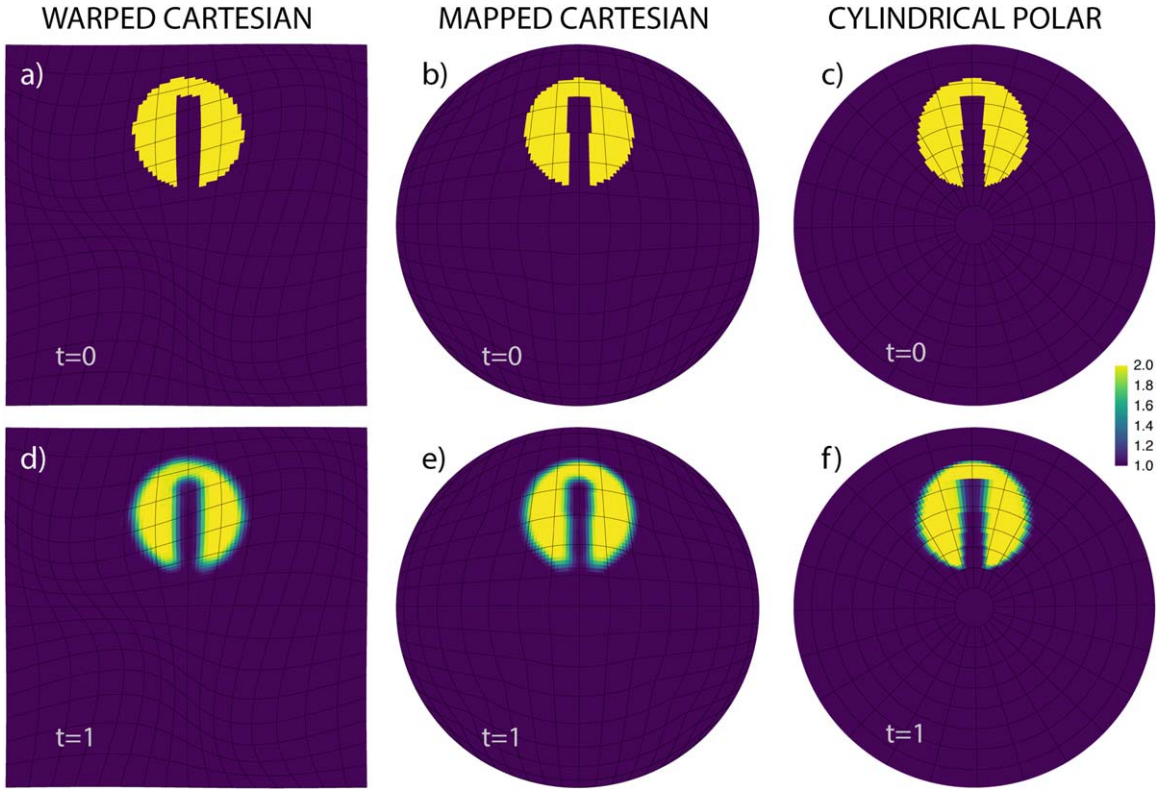


Figure 15. The circular advection results using three different curvilinear grids.

with the seventh-order reconstruction method preserves the shape of the slotted cylinder reasonably well even in nonuniform non-orthogonal grid geometries. The use of high-order reconstruction schemes is essential for the GAMERA code to resolve substantial flow structures with a relatively small amount of computational cells compared to the commonly used second-order TVD schemes, especially when the grid geometry varies significantly in the computation space. Figure 16 shows the 2D circular advection simulation using a second-order TVD scheme and the PDM scheme with the seventh-order reconstruction method.

The top left panel of Figure 16 shows the 2D circular advection simulation results at $t = 1.0$ using the second-order TVD scheme (minmod limiter) with 1024×1024 cells. The

computation grid is the same distorted Cartesian as in Figure 15(a), which is nonuniform and non-orthogonal. The top right panel of Figure 16 shows the corresponding distribution of the slotted cylinder using the GAMERA scheme with a seventh-order reconstruction. The comparisons between the two simulations suggest that qualitatively, the spatial distribution of the slotted density in the second-order TVD scheme resembles that in the seventh-order PDM scheme. The comparison is more quantitative in the line cut profiles of the slotted density, as is shown in the bottom panel of Figure 16. The line cut profiles are taken right through the center of the slotted cylinder at $y = 0.75$. The comparisons between the two cut profiles suggest that in order to get a similar density gradient in the simulation using the PDM scheme with

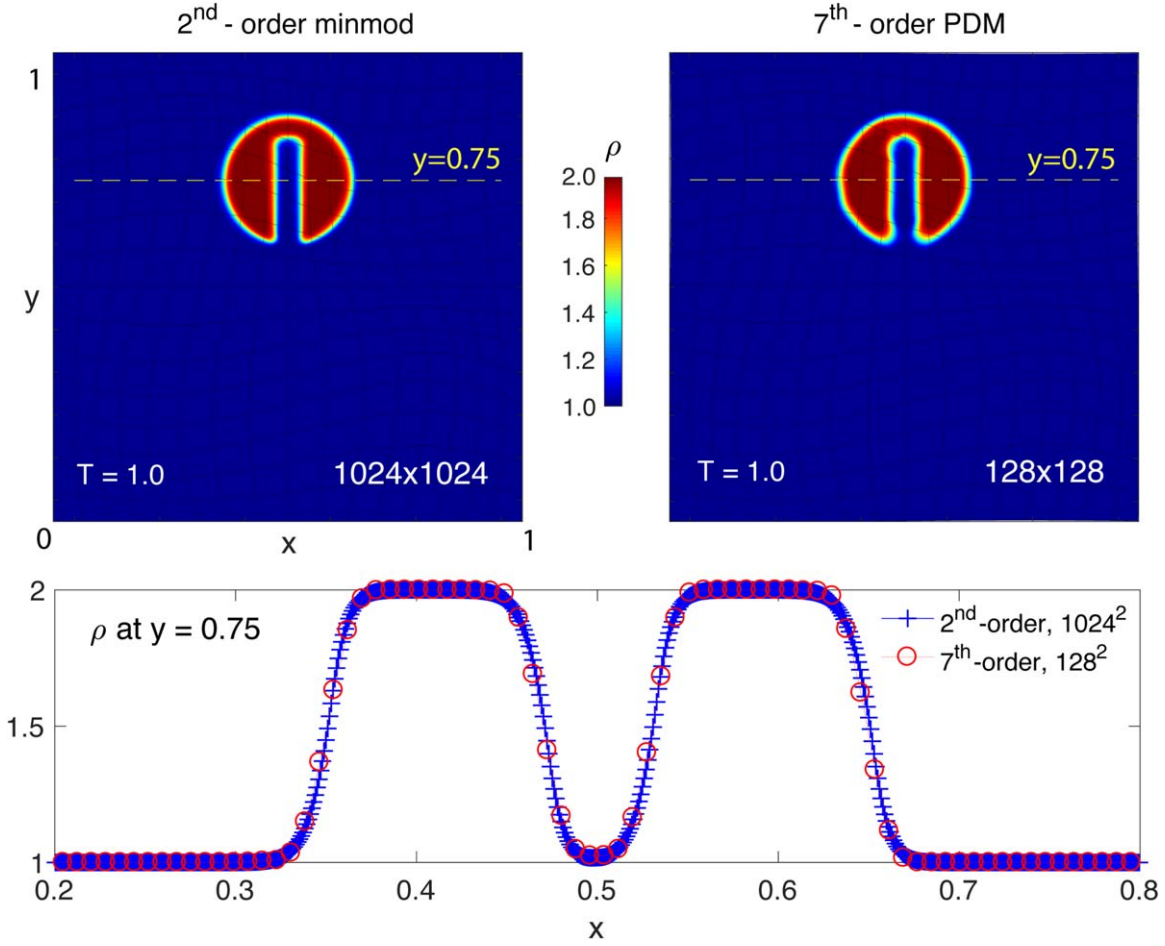


Figure 16. The circular advection results using three different curvilinear grids.

128 × 128 cells, 1024 × 1024 cells are needed while using the second-order TVD scheme, which is 64 × 8 times more computations per unit simulation time for the 2D test problem. Given that the computational cost for the seventh-order PDM scheme is about 24% more than that of a second-order TVD scheme, the total computational cost for the second-order TVD scheme is more than two orders of magnitude higher than the seventh-order PDM scheme. For 3D simulations, the difference in computation time is even more.

4.2. Field-loop Advection

The field-loop advection test consists of the advection of a circular magnetic field loop by a constant initial velocity in a two-dimensional periodic simulation domain. The initial conditions used in this study are adapted from Stone et al. (2008). The simulation domain is a 2D Cartesian box with $-L_x \leq x \leq L_x$, $-L_y \leq y \leq L_y$. The plasma flow is inclined at 30° to the horizontal direction, ensuring that the numerical fluxes in the x and y directions are different, which makes the test truly multidimensional. In order to test the effectiveness of the MHD solver in non-orthogonal grid geometries, three additional sets of field-loop advection simulations are performed using the Cartesian grid with an increasing level of distortion, in which the numerical fluxes in the x and y directions are different regardless of the direction of the flow. The distorted Cartesian grids used in this section are generated

using the following mapping functions:

$$x_{i,j} = L_x \left(2 \left[\frac{i}{N_i} + w_0 \sin\left(\frac{i\pi}{N_i}\right) \sin\left(\frac{j\pi}{N_j}\right) \right] - 1 \right), \quad (102)$$

$$y_{i,j} = L_y \left(2 \left[\frac{j}{N_j} + w_0 \sin\left(\frac{i\pi}{N_i}\right) \sin\left(\frac{j\pi}{N_j}\right) \right] - 1 \right), \quad (103)$$

where $L_x = 1/\sin(\pi/3)$, $L_y = 1/\cos(\pi/3)$, and w_0 is the parameter that determines the amount of grid distortion. The integers $i = 1, 2, \dots, N_i$ and $j = 1, 2, \dots, N_j$ are the grid indices in the μ and ν directions, respectively, where N_i and N_j are the total number of cells in each direction. A similar test for this magnetic field-loop advection using much more complicated warped gridding schemes can be found in Zilhao & Noble (2014).

The initial plasma density ρ and thermal pressure P are both 1.0, with $\gamma = \frac{5}{3}$. The magnitude of the initial flow velocity is 1.0, with $v_x = \cos(\pi/6)$, $v_y = \sin(\pi/6)$, and $v_z = 0$. The magnetic field loop is initialized using the following vector potential $\mathbf{A} = (A_x, A_y, A_z)$:

$$\begin{aligned} A_x &= A_y = 0, \\ A_z &= \text{MAX}([A_0(R_0 - r)], 0), \end{aligned} \quad (104)$$

where A_0 is the magnitude of the field loop, R_0 is the center of the loop, and $r = \sqrt{x^2 + y^2}$. To ensure that the magnetic field

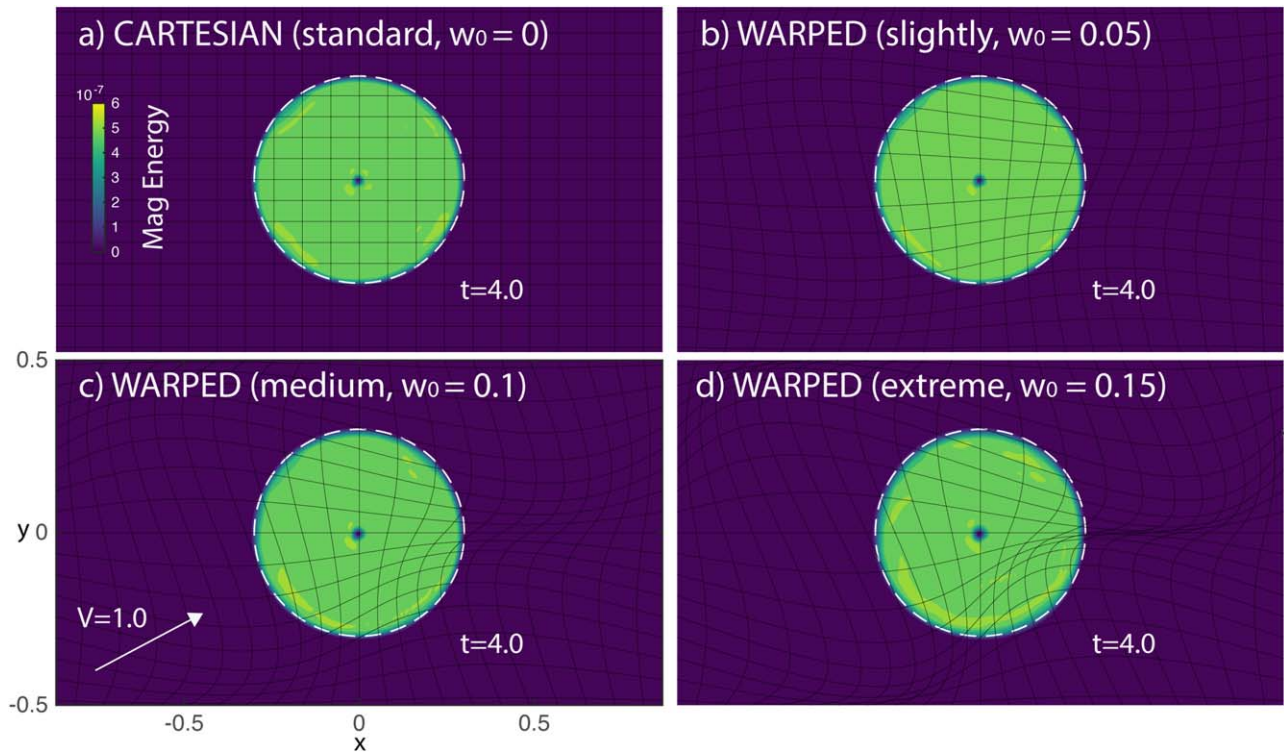


Figure 17. The spatial distribution of magnetic energy at $t = 4$ from four curvilinear grids. An animation of the field-loop advection simulations is available. The animation shows the four curvilinear grids progressing from $t = 0.04$ to 4. (An animation of this figure is available.)

loop is in quasi magnetostatic equilibrium ($\beta \gg 1$), we use $A_0 = 1.0 \times 10^{-3}$ for the loop with $R_0 = 0.3$. Face-centered magnetic fluxes $\Phi_{i,j,k}$ are computed using the face-integrated form of $\nabla \times \mathbf{A}$ to guarantee $\nabla \cdot \mathbf{B} = 0$ initially. Note that the vector potential in Equation (104) has a discontinuous first derivative. Thus, the initial condition has a line current at the center of the loop and a surface return current. These non-smooth currents make the simulation more difficult and provide excellent tests for the effectiveness of the MHD solver handling non-orthogonal curvilinear geometries.

Figure 17 shows the spatial distributions of magnetic energy ($\frac{1}{2}B^2$) at $t = 4.0$ from four sets of simulations using the same initial conditions but different computational grids. At $t = 4.0$, the field loop has been advected around the grid twice. The grid resolution used in each simulation is 256×128 . The boundary of the magnetic field loop ($r = 0.3$) at $t = 0$ is shown using dashed white contours. Figure 17(a) shows the spatial distribution of magnetic energy at $t = 4.0$ using the standard Cartesian grid as a reference, while Figures 17(b)–(d) show the corresponding simulation results in the distorted Cartesian grids with $w_0 = 0.05, 0.1, 0.15$, respectively. The comparisons in Figure 17 show that the spatial distributions of the magnetic energy at $t = 4$ from distorted Cartesian grids resemble those in the standard Cartesian grid, without noticeable distortions or asymmetries in the shape/boundary of the field loop, even using an extremely distorted grid ($w_0 = 0.15$) as shown in Figure 17(d).

Figure 18(a) shows the decay of the volume-integrated magnetic energy as a function of simulation time derived from the four sets of simulations. In the standard Cartesian case, after crossing the simulation box twice ($t = 4.0$), the total magnetic

energy decreases to approximately 98.7% of the original value. The simulations using the distorted Cartesian grids exhibit similar behaviors in the decay of the magnetic energy, but with decreasing final magnetic energy around 98.5%, 98.2%, and 97.7% of the original value as the distortion parameter w_0 increases from 0.05 to 0.15, respectively. As the grid becomes more distorted (from case (b) to case (d)), the rate of magnetic energy decay varies as a function of location, because the field-loop samples grid resolution nonuniformly when it advects along the diagonal. An animation of the field-loop advection simulations in four different grids is available for Figure 17.

Figure 18(b) shows the decay of the total magnetic energy with time using six different orders of reconstruction. The grid geometries used in the six field-loop simulations were all standard Cartesian ($w_0 = 0$) with 256×128 cells. By the time the magnetic field loop crossed the simulation box twice ($t = 4.0$), the total magnetic energy decreased with the order of reconstruction used in the solver. The total magnetic energy in the second-order reconstruction method is about 88.5%, while in the seventh-order reconstruction method, it is about 98.1%. Although the total magnetic energy at the end of the simulation using a second-order reconstruction only degraded approximately 10%, the spatial distribution is significantly distorted compared to the high-order reconstruction schemes (seventh- and eighth-order), which is shown in Figure 19. As the order of reconstruction decreases from 8 to 2, the thickness of the edge of the magnetic field loop increases significantly, which is a consequence of the increased numerical “diffusion width” with decreasing order of reconstruction, as analyzed in detail in Appendix D.

For the 2D MHD test simulations, the additional computing cost of using the seventh-order reconstruction scheme is

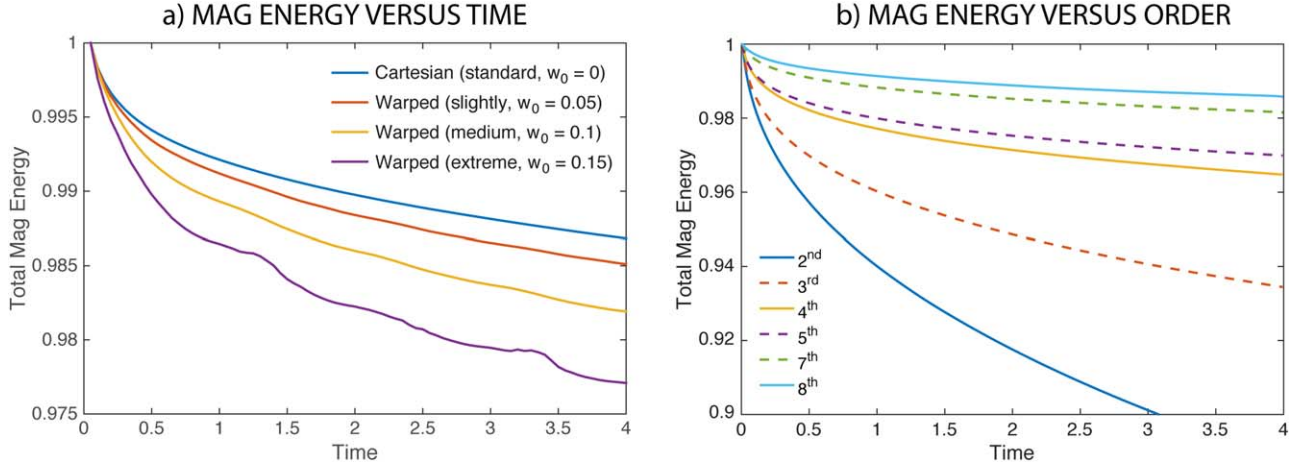


Figure 18. (a) The decay of the total magnetic energy as a function of time derived from the four field-loop advection simulations using different grids. (b) The decay of the total magnetic energy as a function of time from six simulations with different orders of reconstruction in Cartesian geometries.

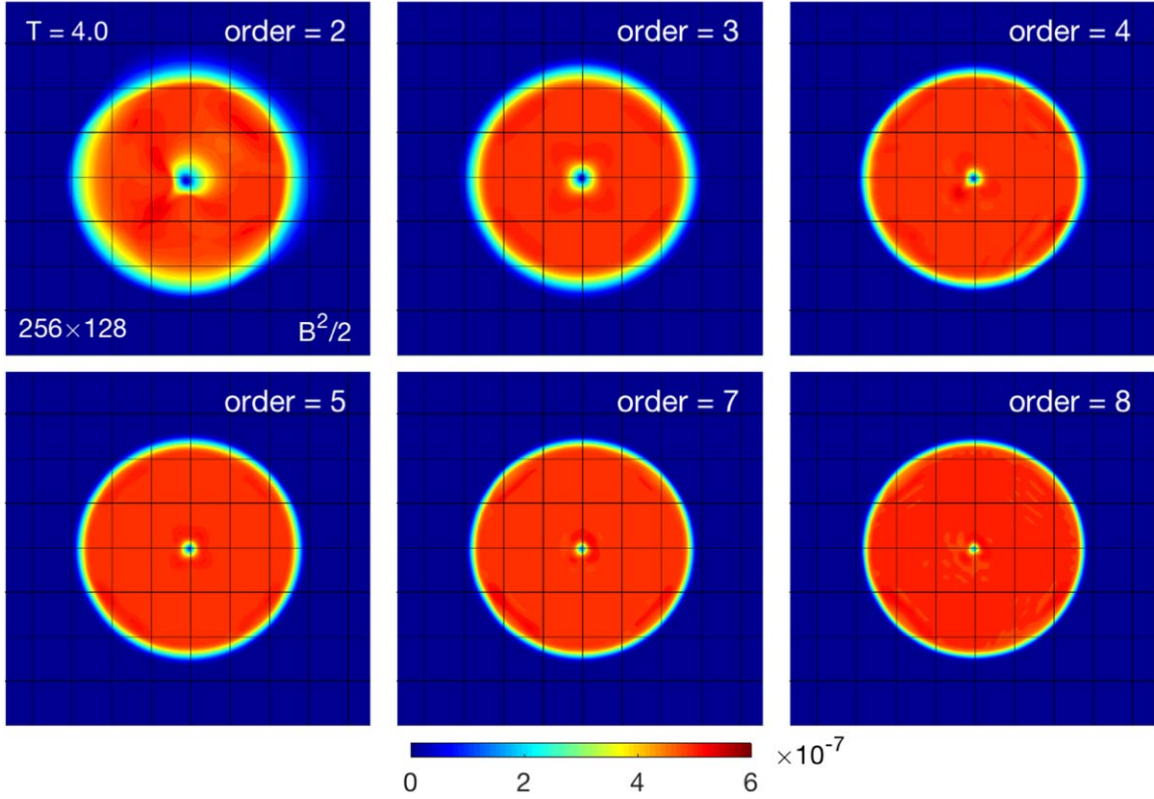


Figure 19. The corresponding spatial distributions of magnetic energy at $t = 4$ using six different orders of reconstruction.

approximately 24% compared to the second-order scheme. When using the eighth-order reconstruction scheme, the additional computational cost is only about 12% more compared to the second-order scheme. For 3D field-loop advection simulations (not shown in this section), the computational costs for going from the second order to the seventh and eighth orders are approximately 23% and 5%, respectively.

4.3. Nonlinear Alfvén Wave

We run the nonlinearly polarized circular Alfvén wave simulation from Tóth (1997) to test the performance of the

numerical MHD schemes in the nonlinear regime. Since the Alfvén wave simulation is smooth, it is also a reasonable test for the convergence of the MHD solver. The simulation domain is chosen to be a 2D Cartesian box with $0 \leq x \leq 1$ and $0 \leq y \leq \frac{1}{\cos \alpha}$, where $\alpha = \frac{\pi}{3}$ is the angle at which the Alfvén wave propagates with respect to the x -axis. Thus, the waves do not propagate along the diagonal of the simulation domain, which guarantees the multidimensional nature of the test simulation by making the numerical flux different in the x and y directions. The initial conditions are $\rho = 1$, $P = 0.1$, $u_{\perp} = 0.1 \sin 2\pi x_{\parallel}$, $B_{\perp} = 0.1 \sin 2\pi x_{\parallel}$, and $u_z = B_z = 0.1 \cos 2\pi x_{\parallel}$ with $\gamma = \frac{5}{3}$ and $x_{\parallel} = (x \cos \alpha + y \sin \alpha)$, where u_{\perp} and B_{\perp} are the components

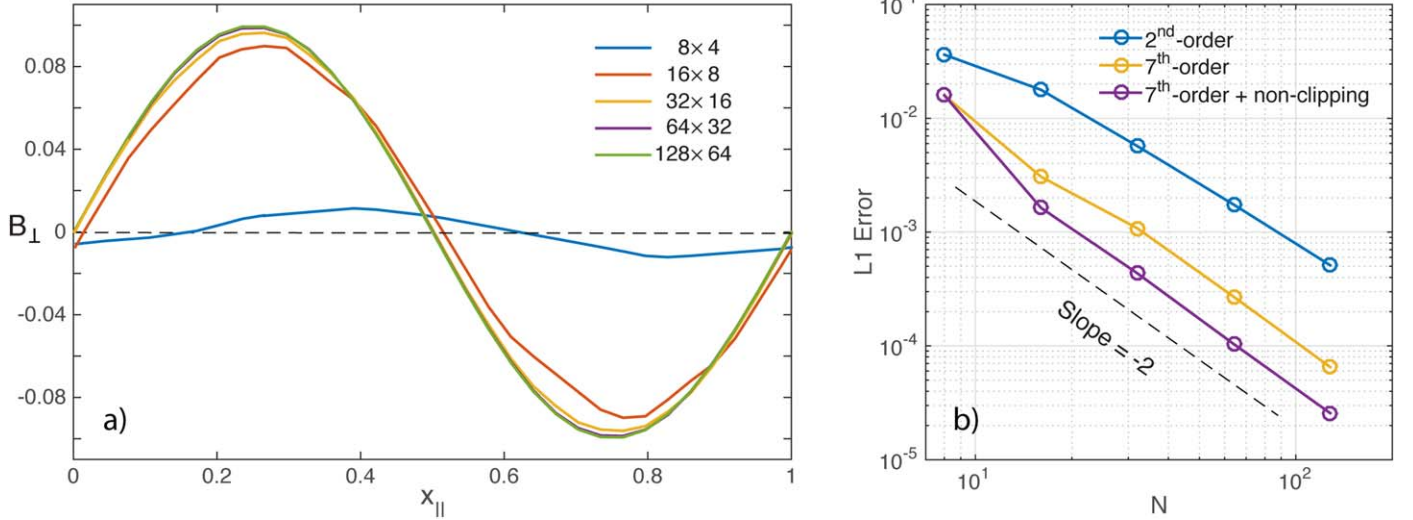


Figure 20. Panel (a) shows the distribution of B_{\perp} vs. x_{\parallel} at $t = 5.0$ in nonlinear Alfvén wave simulations, computed from five runs with increasing resolution. Panel (b) shows the average $L1$ error as a function of grid resolution N .

of the velocity and magnetic field perpendicular to the wave vector. The B_{\parallel} and B_{\perp} components are calculated using cell-centered B_x and B_y via $B_{\perp} = B_y \cos \alpha - B_x \sin \alpha$, and $B_{\parallel} = B_x \cos \alpha + B_y \sin \alpha$. The initial magnetic fluxes Φ are calculated using the following vector potential A :

$$A_x(x, y, z) = B_{\parallel 0} \sin \alpha z \quad (105)$$

$$A_y(x, y, z) = -B_{\parallel 0} \cos \alpha z + \frac{0.1}{2\pi \cos \alpha} \times \sin[2\pi(x \cos \alpha + y \sin \alpha)] \quad (106)$$

$$A_z(x, y, z) = \frac{0.1}{2\pi} \cos[2\pi(x \cos \alpha + y \sin \alpha)], \quad (107)$$

where $B_{\parallel 0} = 1.0$ is the strength of the initial magnetic field parallel to the wave vector.

Figure 20(a) shows the simulated B_{\perp} component along the direction of wave propagation x_{\parallel} at $t = 5.0$, using the default scheme with the seventh-order spatial reconstruction and the PDM limiter. Six grid resolutions with $N \times \frac{N}{2}$ cells are used to test the convergence of the solution at $t = 5.0$ by increasing the grid parameter N from 8 to 256. When the grid resolution is 64×32 and greater, the solution of B_{\perp} along x_{\parallel} starts to converge to the analytical solution $B_{\perp} = 0.1 \sin 2\pi x_{\parallel}$. Figure 20(b) shows the average $L1$ error norm as a function of grid size N using two different reconstruction methods (second- and seventh-order reconstruction). We find a second-order convergence rate based on this smooth but nonlinear Alfvén wave simulation, regardless of the size of the reconstruction stencil, suggesting that the formal order of convergence for the MHD solver is two. This is expected, since no high-order quadrature methods are used in the evaluation of face fluxes in the finite-volume solver as discussed in Section 3.4. However, the formal order of convergence (second order) does not mean that the high-order reconstruction (seventh order and above) is not necessary in the solver. In fact, it is important to note that when using the default seventh-order reconstruction scheme, the magnitude of the average $L1$ error is approximately a factor of 10 lower than that from a second-order reconstruction scheme, suggesting that using high-order reconstructions for achieving highly accurate

solutions with a moderate number of computational cells is necessary. This reduction in average $L1$ error is consistent with the comparisons in Figure 15, which show significant improvement in preserving the shape of initial density structures. Note that when the non-clipping option is switched on, the average $L1$ error is further reduced by a factor of ~ 3 , due to the preservation of local extrema.

Figure 21 compares the spatial distributions of B_{\perp} at $t = 5.0$ from four sets of nonlinear Alfvén wave simulations using exactly the same initial and boundary conditions in Cartesian geometries with different levels of distortion. The grid size used in the test simulations is 256×128 . Figure 21(a) is from a standard orthogonal Cartesian grid, and Figures 21(b)–(d) are from distorted non-orthogonal Cartesian grids defined in Equations (102)–(103) with $w_0 = 0.05, 0.1$, and 0.15 , respectively. Compared to the Cartesian case in Figure 21(a), the simulated wavefronts indicated by B_{\perp} in the distorted non-orthogonal grids are perpendicular to the direction of propagation, without distortions introduced by the non-orthogonal grids. Figure 22 shows the comparison on the perpendicular component of the magnetic field (B_{\perp}) along the wave vector derived from the four test simulations with different grids. The cut profiles were taken at the center of each simulation domain for one parallel wavelength. These comparisons shown in Figures 21 and 22 also demonstrate the capability of the numerical schemes handling smooth MHD solutions in the nonlinear regime when using non-orthogonal grids.

4.4. Orszag–Tang Vortex

We use the Orszag–Tang vortex simulation to test how robust the code is for handling the formation and interaction of nonlinear MHD shocks in non-orthogonal, curvilinear geometry. The Orszag–Tang vortex is a well-known two-dimensional problem for testing the transition to supersonic MHD turbulence, which is a very common test of numerical MHD schemes used in many previous studies as a basis for consistent comparison of codes. In this test, we focus on the effectiveness of the numerical schemes in handling non-orthogonal grids and compare the results in non-orthogonal geometries with those from a standard uniform Cartesian grid. For simulating the

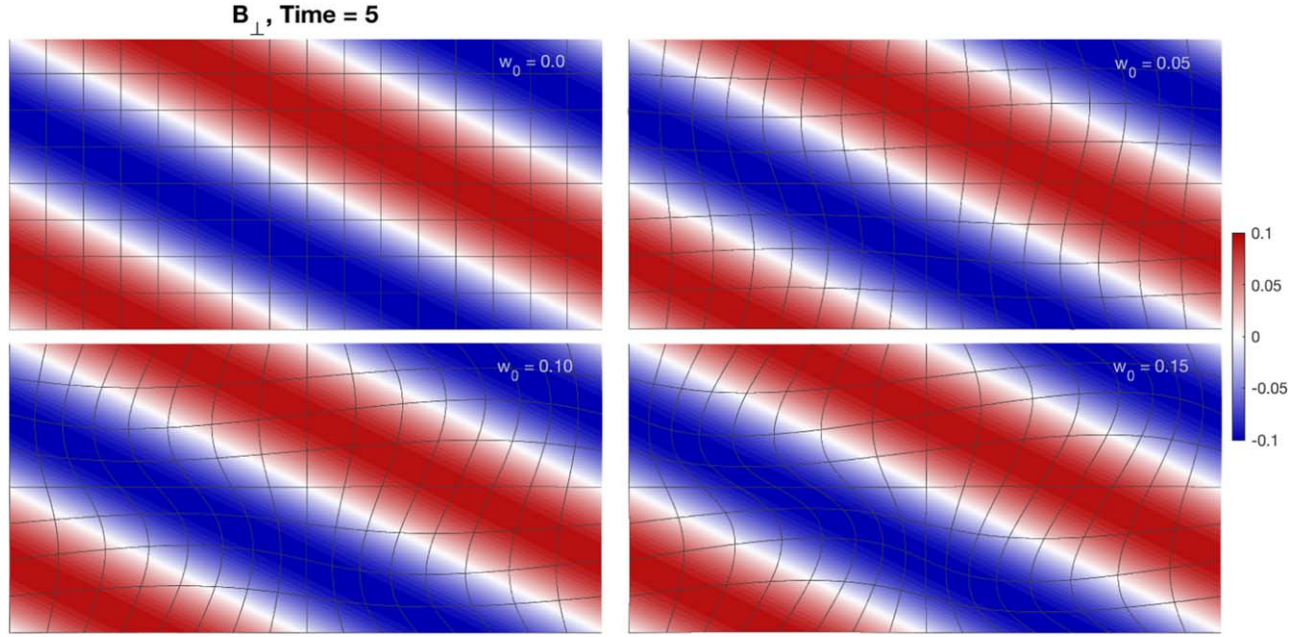


Figure 21. The spatial distribution of B_{\perp} at $t = 5.0$ derived from four nonlinear Alfvén wave simulations of Cartesian geometries with different levels of distortion indicated by the w_0 parameter in Equations (102)–(103). The grid resolution used in each simulation is 256×128 cells.

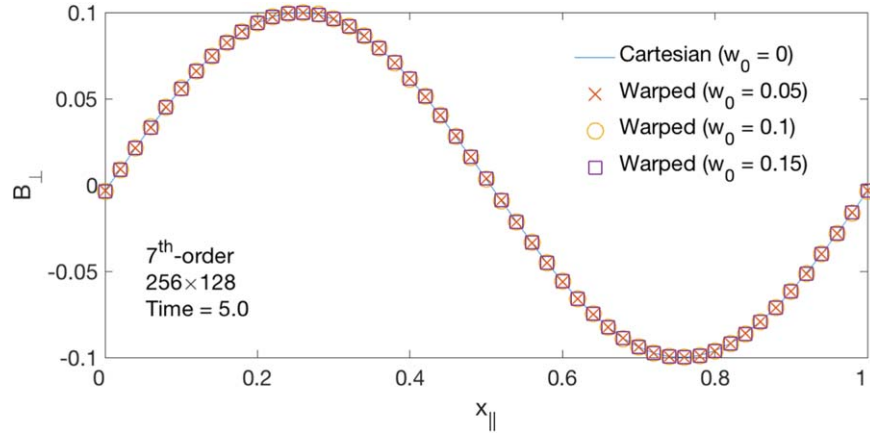


Figure 22. The spatial distribution of B_{\perp} at $t = 5.0$ along the wave vector derived from the four simulations with different levels of grid distortion.

Orszag–Tang vortex, we use a 2D square simulation domain with $0 \leq x \leq 1$ and $0 \leq y \leq 1$. The initial density is $\rho = \frac{25}{36}\pi$ and the initial pressure is $P = \frac{5}{12}\pi$ distributed uniformly in the simulation domain, with $\gamma = \frac{5}{3}$. The initial velocities are set as $v_x = -\sin(2\pi y)$ and $v_y = \sin(2\pi x)$. The magnetic fluxes Φ are initialized using the following vector potential function A :

$$\begin{aligned} A_x &= A_y = 0, \\ A_z &= B_0 \left(\frac{1}{4\pi} \cos(4\pi x) + \frac{1}{2\pi} \cos(2\pi y) \right) \end{aligned} \quad (108)$$

where $B_0 = 1.0$. The above vector potential function gives $B_x = -B_0 \sin(2\pi y)$ and $B_y = B_0 \sin(4\pi x)$ initially. The boundary conditions are periodic everywhere. All of the test simulations shown in Figure 23 have a computational grid with 512×512 cells. The reconstruction method is the default seventh-order scheme without the non-clipping option switched on.

Figure 23(a) shows the simulated spatial distributions of the plasma pressure at simulation time $t = 0.48$ in a uniform Cartesian grid, when the MHD shocks start to interact near the center of the simulation domain. Figures 23(b)–(d) show the corresponding spatial distributions of plasma pressure at the same simulation time using three sets of non-orthogonal, distorted Cartesian grids as defined in Equations (102)–(103) with increasing w_0 . Although these comparisons are less quantitative, they do suggest that the simulated distributions of plasma pressure are not significantly affected by the choice of the computational grid. Both smooth structures and shocks at $t = 0.48$ using nonuniform non-orthogonal grids are remarkably similar to those in Figure 23(a) using a uniform Cartesian grid. The effectiveness of the MHD solver is also illustrated in more quantitative comparisons of using line profiles. Figure 24 shows the comparisons of the simulated plasma pressure profiles (at $t = 0.48$) along the y direction, with $x = 0.5$. Although the simulated pressure exhibits some minor

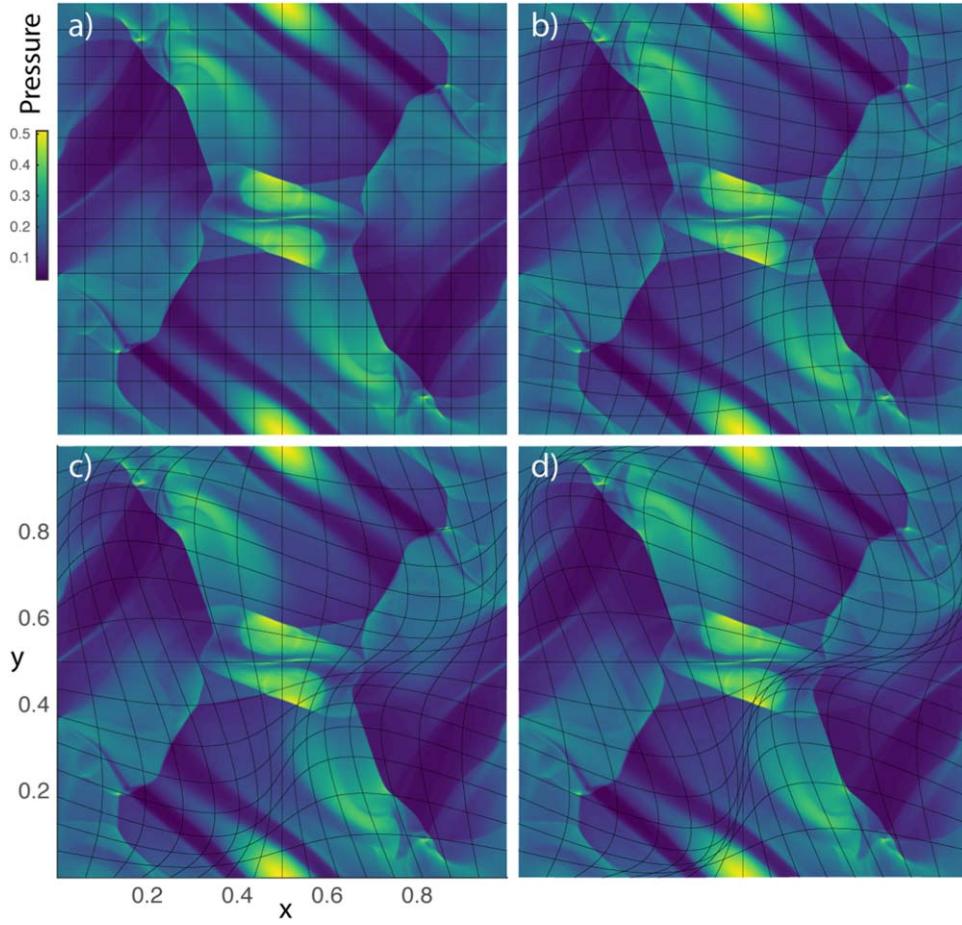


Figure 23. The spatial distribution of plasma pressure P at $t = 0.48$ in four Orszag–Tang simulations using different grids. Each simulation has 512×512 cells. An animation of the Orszag–Tang vortex simulation using the four different grid geometries is available. The animation proceeds from $t = 0.005$ to 0.50 .

(An animation of this figure is available.)

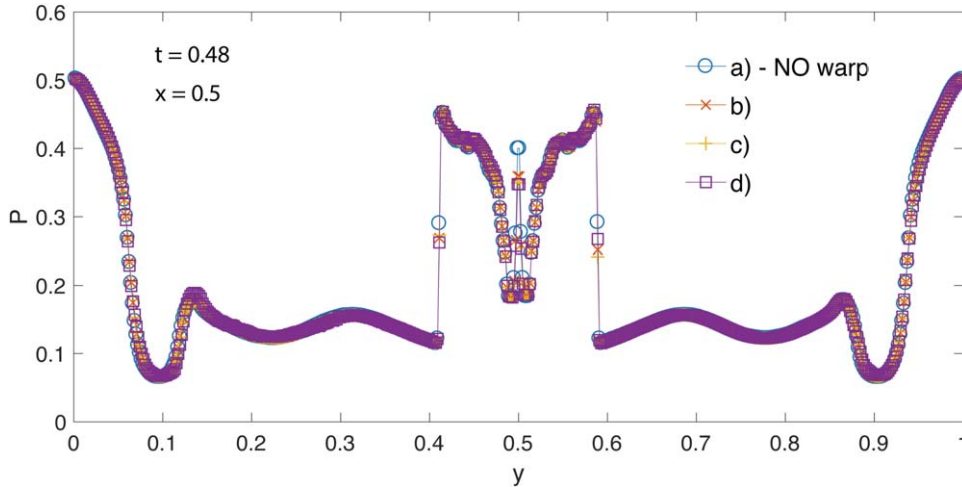


Figure 24. The line profiles of plasma pressure at $x = 0.5$ from four Orszag–Tang simulations using different grids, taken from same simulation snapshots with $t = 0.48$.

differences at the shock transition regions, possibly due to the inhomogeneity of the grid, very little difference occurs in the smooth regions. The Orszag–Tang vortex simulations demonstrate the capability of the GAMERA code to handle MHD

shock propagation and interaction, without the requirement of the orthogonality of the computation geometry. An animation of the Orszag–Tang vortex simulation using the four different grid geometries is shown in Figure 23.

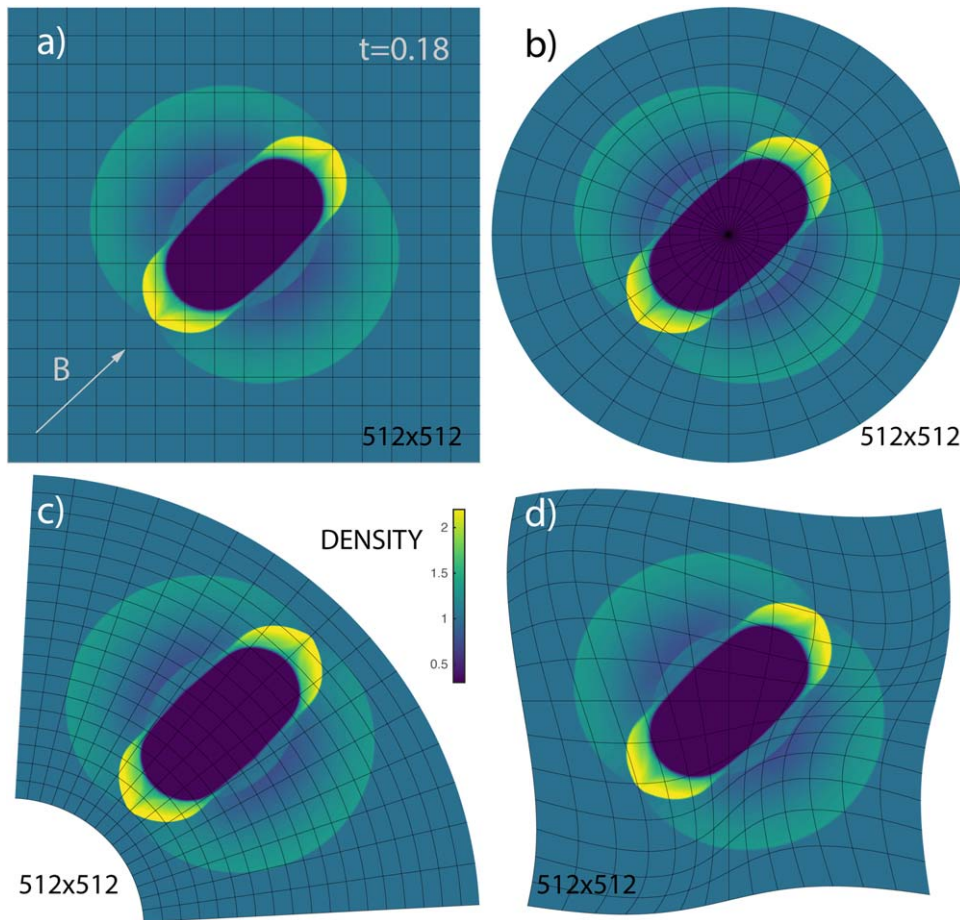


Figure 25. The spatial distribution of plasma pressure P at $t = 0.18$ in four MHD spherical blast-wave simulations using different grids. Each simulation has 512×512 cells. An animation of the plasma pressure in the MHD blast-wave simulation in the four different grids is available. The animation proceeds from $t = 0.002$ to 0.18 .

(An animation of this figure is available.)

4.5. Spherical Blast Wave

We use MHD blast-wave simulations to test the robustness the numeric schemes in handling multidimensional strong MHD shocks and rarefactions. In a two-dimensional simulation domain in the x - y plane, the initial plasma has uniform density $\rho = 1$, thermal pressure $P = 0.1$, and zero velocity $\mathbf{u} = 0$, with $\gamma = \frac{5}{3}$. Within the spatial region $r = \sqrt{x^2 + y^2} < 0.1$, the thermal pressure P is set to 10.0 (that is, 100 times greater than the ambient plasma pressure). The initial magnetic field is along the diagonal in the x - y plane with $B_x = \frac{1}{\sqrt{2}}$, $B_y = \frac{1}{\sqrt{2}}$ and $B_z = 0$.

We use four different computational grids for the MHD blast-wave simulation with the same initial conditions. These grids include a standard Cartesian grid (Figure 25(a)), a cylindrical polar grid (Figure 25(b)), a quarter of a ring (Figure 25(c)), and a distorted Cartesian grid (Figure 25(d)) as used in the Orszag–Tang simulations. Each grid has a resolution of 512×512 cells with the center of the blast wave located at the origin. The Cartesian grid is used as a reference, while the non-Cartesian grids are used to test the propagations of strong MHD shocks and rarefaction in non-orthogonal curvilinear geometries. The cylindrical polar grid is singular at the pole, which is treated using a conservative averaging-reconstruction technique developed by Zhang et al. (2019b)

without changing the numerical schemes of the MHD solver. An animation of the MHD blast-wave simulation in the four different grids is shown in Figure 25. As shown in the animation, at early simulation times (e.g., $t < 0.05$), the outgoing blast waves are approximately spherical and show no grid alignment effects due to non-orthogonal curvilinear geometries. The results shown in Figure 25 are the spatial distributions of the plasma density at $t = 0.18$. Although the spherical blast-wave simulations are not very quantitative, they show that the spatial distributions of plasma density in the blast-wave simulation at a later time are not sensitive to the choice of the curvilinear grid. In the simulation results shown in Figure 25, both smooth structures and shocks in the non-Cartesian geometries (Figures 25(b)–(d)) are remarkably similar to those in the reference Cartesian grid (Figure 25(a)) without noticeable differences. Note that the Richtmyer–Meshkov instability is suppressed by the strong magnetic field, and no fingers are evident in the interior of the field-aligned bubble, as are those in a hydrodynamic blast-wave simulation.

5. Summary

The GAMERA code is a reinvention of the LFM MHD kernel with significant upgrades in both numerical schemes and software implementation. The improvements in numerical schemes include (1) 12th-order grid metric calculations using

Gaussian quadrature, (2) a high-order upwind reconstruction method, (3) the PDM limiter with an extremum-preserving option, and (4) background magnetic field splitting for very low- β conditions. The ring-average technique for spherical axis singularity described by Zhang et al. (2019b) is also implemented in the GAMERA code for high-resolution simulations in general geometry with axis singularities. The improvements in software implementation include (1) modern Fortran implementation with minimum external library dependence, (2) core computational routines that act on vector-length aligned blocks of memory, (3) extensive use of OpenMP threading within a socket allowing MPI ranks to be distributed across sockets, and (4) High-performance Computing (HPC)-friendly data structures, namely the use of large contiguous arrays and contiguous memory access patterns. The GAMERA code is designed to be easily applicable to multidimensional MHD flow simulations in non-orthogonal curvilinear grids adapted to specific conditions. Current applications of the GAMERA code include global simulations of planetary magnetospheres, the inner heliosphere and solar wind, and local simulations of basic plasma physics applications, e.g., current sheets.

This paper provides a comprehensive description of the numerical recipes used in the GAMERA code. Our hope in the development of this paper has been to give some context for the key questions and solutions that need to be addressed in order to design a large-scale simulation code for MHD problems. The numerical recipes described in this paper are useful as a reference, but they also show that, when compared with other codes, these recipes are not the same. The nuances of disparate application areas will lead to different codes that excel in their tailored regimes. Although the numerical considerations of the LFM, inherited by GAMERA, are tailored to heliospheric environments, we have designed GAMERA to be more general. We expect GAMERA to be a useful tool in any application where geometric flexibility is important.

This research was supported by the National Center for Atmospheric Research (NCAR) and the Johns Hopkins University Applied Physics Laboratory Independent Research & Development funds. B.Z. was supported by the HKU Seed Fund for Basic Research (project 201807159001) and RGC Early Career Scheme (project 27302018). The authors would like to acknowledge the high-performance computing support from Cheyenne (doi:10.5065/D6RX99HX) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation (NSF). M.W. was serving at the NSF during the production of the this paper. Any opinion, findings, or conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Appendix A Normalization of the Ideal MHD Equations

The mass and momentum equations in SI units follow as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (109)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u} + \bar{I} P) - \frac{1}{\mu_0} \mathbf{j} \times \mathbf{B} = 0, \quad (110)$$

where $\mu_0 = 4\pi \times 10^{-7} \text{ T} \cdot \text{m/A}$. To normalize the MHD variables in the above equations, define

$$\rho = \rho_0 \tilde{\rho}, \quad (111)$$

$$\mathbf{u} = u_0 \tilde{\mathbf{u}}, \quad (112)$$

$$P = P_0 \tilde{P}, \quad (113)$$

$$\mathbf{B} = B_0 \tilde{\mathbf{B}}, \quad (114)$$

where ρ_0 , u_0 , P_0 , and B_0 are the normalization constants for mass, velocity, pressure, and magnetic field, respectively. $\tilde{\rho}$, $\tilde{\mathbf{u}}$, \tilde{P} , and $\tilde{\mathbf{B}}$ are the corresponding dimensionless variables evolved by the MHD solver. For the time and spatial coordinates, let $t = t_0 \tilde{t}$ and $\mathbf{x} = x_0 \tilde{\mathbf{x}}$ where \tilde{t} and $\tilde{\nabla}$ are the dimensionless temporal and spatial differential operators:

$$\frac{\partial}{\partial t} = \frac{1}{t_0} \frac{\partial}{\partial \tilde{t}}, \quad (115)$$

$$\nabla = \frac{1}{x_0} \tilde{\nabla}. \quad (116)$$

First substitute (111) and (112) into the mass Equation (109):

$$\begin{aligned} \frac{1}{t_0} \frac{\partial \rho_0 \tilde{\rho}}{\partial \tilde{t}} + \frac{1}{x_0} \tilde{\nabla} \cdot (\rho_0 \tilde{\rho} u_0 \tilde{\mathbf{u}}) \\ = 0 \longrightarrow \frac{x_0}{t_0 u_0} \frac{\partial \tilde{\rho}}{\partial \tilde{t}} + \tilde{\nabla} \cdot (\tilde{\rho} \tilde{\mathbf{u}}) = 0. \end{aligned} \quad (117)$$

If we choose the following normalization between time and velocity:

$$t_0 = \frac{x_0}{u_0}, \quad (118)$$

the dimensionless mass equation is written as

$$\frac{\partial \tilde{\rho}}{\partial \tilde{t}} + \tilde{\nabla} \cdot (\tilde{\rho} \tilde{\mathbf{u}}) = 0. \quad (119)$$

Next, substitute (111)–(114) into the momentum Equation (110), and given that $x_0 = u_0 t_0$ from the normalized mass equation,

$$\begin{aligned} \frac{\rho_0 u_0}{t_0} \frac{\partial \tilde{\rho} \tilde{\mathbf{u}}}{\partial \tilde{t}} + \frac{1}{x_0} \tilde{\nabla} \cdot (\rho_0 u_0^2 \tilde{\rho} \tilde{\mathbf{u}} \tilde{\mathbf{u}} + \bar{I} P_0 \tilde{P}) \\ - \frac{B_0^2}{\mu_0 x_0} \tilde{\nabla} \times \tilde{\mathbf{B}} \times \tilde{\mathbf{B}} = 0 \end{aligned} \quad (120)$$

$$\begin{aligned} \implies \frac{\partial \tilde{\rho} \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\nabla} \cdot \left(\tilde{\rho} \tilde{\mathbf{u}} \tilde{\mathbf{u}} + \bar{I} \tilde{P} \frac{P_0}{\rho_0 u_0^2} \right) \\ - \frac{B_0^2}{\mu_0 \rho_0 u_0^2} \tilde{\nabla} \times \tilde{\mathbf{B}} \times \tilde{\mathbf{B}} = 0. \end{aligned} \quad (121)$$

The above equation gives the following normalization for the plasma pressure P_0 and the magnetic field B_0 :

$$P_0 = \rho_0 u_0^2 \quad (122)$$

$$B_0^2 = \mu_0 \rho_0 u_0^2. \quad (123)$$

Thus, the dimensionless momentum equation is written as

$$\frac{\partial \tilde{\rho} \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\nabla} \cdot (\tilde{\rho} \tilde{\mathbf{u}} \tilde{\mathbf{u}} + \bar{I} \tilde{P}) - \tilde{\nabla} \times \tilde{\mathbf{B}} \times \tilde{\mathbf{B}} = 0. \quad (124)$$

It is straightforward to show that with the normalization relations in Equations (118), (122), and (123), the

dimensionless plasma energy equation and Faraday's law are written as

$$\frac{\partial \tilde{E}_P}{\partial \tilde{t}} = -\tilde{\nabla} \cdot [\tilde{\mathbf{u}}(\tilde{E}_P + \tilde{P})] - \tilde{\mathbf{u}} \cdot \tilde{\nabla} \cdot \left(\frac{\tilde{B}^2}{2} \mathbf{I} - \tilde{\mathbf{B}}\tilde{\mathbf{B}} \right) \quad (125)$$

$$\frac{\partial \tilde{\mathbf{B}}}{\partial \tilde{t}} = \tilde{\nabla} \times \tilde{\mathbf{u}} \times \tilde{\mathbf{B}}. \quad (126)$$

where $\tilde{E}_P = \frac{1}{2}\tilde{\rho}\tilde{u}^2 + \frac{\tilde{P}}{\gamma-1}$ is the normalized plasma energy. To summarize, Equations (119), (124), (125), and (126) form a set of dimensionless MHD equations with the following normalizations:

$$t_0 = \frac{x_0}{u_0}, \quad (127)$$

$$P_0 = \rho_0 u_0^2, \quad (128)$$

$$B_0 = \sqrt{\mu_0 \rho_0} u_0. \quad (129)$$

Appendix B The Grid Metric Calculations

As described in Section 3.1, the *primary* grid is made up of the cell corner coordinates $\mathbf{x}_{i\pm\frac{1}{2},j\pm\frac{1}{2},k\pm\frac{1}{2}}$. Thus, the corresponding cell center positions $\mathbf{x}_{i,j,k}^C$ for cell (i, j, k) as shown in Figure 1 are computed using the eight cell corners forming a hexahedron as

$$\begin{aligned} \mathbf{x}_{i,j,k}^C = & \frac{1}{8}(\mathbf{x}_{i-\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + \mathbf{x}_{i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} \\ & + \mathbf{x}_{i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \mathbf{x}_{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \\ & + \mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} \\ & + \mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}). \end{aligned} \quad (130)$$

The calculated cell center location $\mathbf{x}_{i,j,k}^C$ should be located in the control volume (i, j, k) . In some cases, reasonably shaped grid designs could possibly put $\mathbf{x}_{i,j,k}^C$ close to a face when the cell is a concave; thus, the numerical methods may not work well due to the extremes in the metric calculations. However, this situation has not occurred in any of the heliophysics applications developed using GAMERA. In the GAMERA code, the face center locations \mathbf{x}^μ , \mathbf{x}^ν , and \mathbf{x}^ζ are used directly in the calculation of the magnetic field vectors. Thus, they are calculated using high-order Gaussian quadrature, which gives better definitions of the “barycenter” of a cell interface especially when the cells are either very distorted or degenerated to tetrahedrons, e.g., spherical grid cells near the axis. For example, when using two-dimensional Gaussian quadratures, the μ -face center locations $\mathbf{x}_{i+\frac{1}{2},j,k}^\mu$ are calculated as

$$\begin{aligned} \mathbf{x}_{i+\frac{1}{2},j,k}^\mu = & \frac{1}{A_{i+\frac{1}{2},j,k}^\mu} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \mathbf{x} \left(\mu = i + \frac{1}{2}, \nu, \zeta \right) d\nu d\zeta \\ = & \frac{1}{A_{i+\frac{1}{2},j,k}^\mu} \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m \mathbf{x}_{n,m}^{\mu=i+1/2} |J_{n,m}^{\nu,\zeta}| \end{aligned} \quad (131)$$

where $A_{i+\frac{1}{2},j,k}^\mu$ is the μ face area at interface $(i + \frac{1}{2}, j, k)$, which is also calculated using Gaussian quadrature. n and m are the

indices of the Gaussian quadrature points in the ν and ζ directions, respectively. w_n and w_m are the corresponding weights at the Gaussian points. $\mathbf{x}_{n,m}$ is the corresponding spatial location of the 2D Gaussian quadrature points with indices (n, m) on the ν - ζ interface, and $|J_{n,m}^{\nu,\zeta}|$ is the corresponding Jacobian of the local coordinate transformation. In the GAMERA code, the quadrature is done using the 12th-order two-dimensional Gaussian quadrature ($N = 12$). The actual implementation of the 12th-order Gaussian quadrature can be found in the example Matlab source code.

Similarly, the face center locations at the ν and ζ interfaces are calculated as

$$\begin{aligned} \mathbf{x}_{i,j+\frac{1}{2},k}^\nu = & \frac{1}{A_{i,j+\frac{1}{2},k}^\nu} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \mathbf{x} \left(\mu, \nu = j + \frac{1}{2}, \zeta \right) d\zeta d\mu \\ = & \frac{1}{A_{i,j+\frac{1}{2},k}^\nu} \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m \mathbf{x}_{n,m}^{\nu=j+1/2} |J_{n,m}^{\zeta,\mu}| \end{aligned} \quad (132)$$

$$\begin{aligned} \mathbf{x}_{i,j,k+\frac{1}{2}}^\zeta = & \frac{1}{A_{i,j,k+\frac{1}{2}}^\zeta} \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \mathbf{x} \left(\mu, \nu, \zeta = k + \frac{1}{2} \right) d\mu d\nu \\ = & \frac{1}{A_{i,j,k+\frac{1}{2}}^\zeta} \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m \mathbf{x}_{n,m}^{\zeta=k+1/2} |J_{n,m}^{\mu,\nu}| \end{aligned} \quad (133)$$

where the integrals are done on corresponding cell faces formed by four corner points (or three corner points when the cell faces are degenerated). Numerical solutions exhibit fewer artifacts in the grids with axis singularities or highly distorted aspect ratios when the “barycenters” of the cell faces defined in Equations (131)–(133) are used.

The face areas $A_{i\pm\frac{1}{2},j,k}^\mu$, $A_{i,j\pm\frac{1}{2},k}^\nu$, and $A_{i,j,k\pm\frac{1}{2}}^\zeta$ are also calculated using the 12th-order two-dimensional Gaussian quadrature based on the four corner points forming the cell face:

$$A_{i+\frac{1}{2},j,k}^\mu = \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} d\nu d\zeta = \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m |J_{n,m}^{\nu,\zeta}| \quad (134)$$

$$A_{i,j+\frac{1}{2},k}^\nu = \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} d\zeta d\mu = \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m |J_{n,m}^{\zeta,\mu}| \quad (135)$$

$$A_{i,j,k+\frac{1}{2}}^\zeta = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} d\mu d\nu = \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_n w_m |J_{n,m}^{\mu,\nu}|. \quad (136)$$

The cell volume $V_{i,j,k}$ is also calculated using a 12th-order three-dimensional Gaussian quadrature based on the hexahedron cell formed by the eight cell corners shown in Figure 1:

$$\begin{aligned} V_{i,j,k} = & \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} d\mu d\nu d\zeta \\ = & \sum_{l=1}^{l=N} \sum_{m=1}^{m=N} \sum_{n=1}^{n=N} w_l w_m w_n |J_{l,m,n}^{\mu,\nu,\zeta}|. \end{aligned} \quad (137)$$

Appendix C The General Form of the PDM Operator

For nonlinear problems, the form of the PDM operator M^+ using a second-order centered reconstruction is written as (Hain 1987)

$$M^+ = m^+ - \frac{1}{2} \text{sign}(v) \text{sign}(\nabla^+) \max[0, |\nabla^+|] \\ - \frac{A}{4} \cdot (1 - \text{sign}(v)) \cdot |\text{sign}(\nabla^+) + \text{sign}(\nabla^{++})| \cdot |\nabla^{++}| \\ - \frac{A}{4} \cdot (1 + \text{sign}(v)) \cdot |\text{sign}(\nabla^-) + \text{sign}(\nabla^+)| \cdot |\nabla^-|, \quad (138)$$

where $m^+ f = \frac{1}{2}(f_i + f_{i+1})$ corresponds to the second-order reconstruction, $\nabla^{++} f = f_{i+2} - f_{i+1}$, $\nabla^+ f = f_{i+1} - f_i$, $\nabla^- f = f_i - f_{i-1}$, and A is the parameter that determines the amount of numerical diffusion. The direction of sweeping is $\text{sign}(v)$; thus, Equation (138) gives both the left and right state at interface $i + \frac{1}{2}$. If $A = 0$, the above operator M^+ becomes the Donor Cell method, which is one of the most diffusive first-order methods. For linear problems, Hain (1987) showed that the monotonicity condition for the PDM parameter A is related to the Courant number N_{CFL} :

$$N_{\text{CFL}} < \frac{2}{2 + A}. \quad (139)$$

In principle, any value of $A > 0$ could be used in the PDM limiter. Larger values of A lead to more aggressive limiting as indicated by Equation (138). However, the allowable Courant number decreases with increasing A , as shown in Equation (139). In practice, we find that there is little improvement in resolving square wave profiles if $A > 4$. Thus, in the GAMERA solver, the default value for the PDM value A is chosen to be 4.0, together with a fixed Courant number $N_{\text{CFL}} = 0.3$ which satisfies Equation (139). The PDM limiter can be extended to use an arbitrary high-order spatial reconstruction instead of using m^+ in Equation (138), as shown in Figure 26. For a high-order interface value f^{HO} reconstructed at $i + \frac{1}{2}$, the left state ($v > 0$) at interface $i + \frac{1}{2}$ is calculated using the following equation:

$$f_{\text{PDM}}^L = f_{\text{HO}}^* - \text{sign}(f_{i+1} - f_i) \cdot \max \\ \times \left[0, |f_{\text{HO}}^* - f_i| - \frac{A}{4} \cdot |f_i - f_{i-1}| \cdot \frac{1}{2} |\text{sign}(f_i - f_{i-1}) + \text{sign}(f_{i+1} - f_i)| \right]. \quad (140)$$

Here, the f_{HO}^* is a “clipped” version of the high-order reconstruction value f_{HO} :

$$f_{\text{HO}}^* = \text{median}(f_i, f_{\text{HO}}, f_{i+1}). \quad (141)$$

It is very straightforward to show that if $A = 0$, the above equation becomes

$$f_{\text{PDM}}^L = f_i, \quad (142)$$

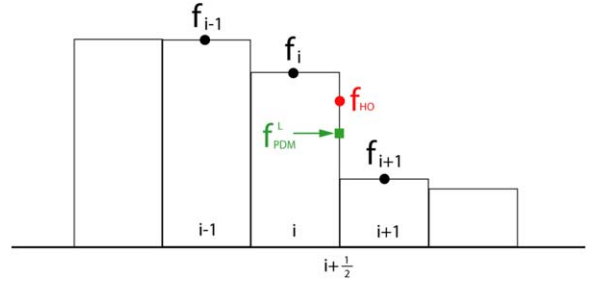


Figure 26. The calculation of the left state at interface $i + \frac{1}{2}$ using the PDM limiter.

which is the first-order Donor Cell method. The numerical diffusion is reduced significantly as A is increased from 0 to 4.0.

Appendix D The Choice of Reconstruction Order

The PDM limiter can be combined with an arbitrarily high-order reconstruction scheme as discussed in Section 3.3.1. In the GAMERA code, the default choice of the reconstruction method is the seventh-order upwind reconstruction (the eighth-order centered reconstruction is the one used in the original LFM, which is also implemented in the GAMERA code for reference). The reason for choosing such high-order reconstruction schemes as the defaults is based on achieving low numerical diffusion with a reasonable amount of computing resource, which is important for 3D global-scale simulations of large space plasma systems such as planetary magnetospheres and the heliosphere. Here, we show two sets of numerical solutions of the linear advection equation to justify the necessity of the high-order reconstruction implemented in the GAMERA code.

D.1. 1D Linear Advection of Four Shapes in a Nonuniform Grid

The 1D linear advection equation solved in this section follows

$$\frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} = 0, \quad (143)$$

where $u = 1.0$ and x is defined on $[-1, 1]$. The computational domain is nonuniform with

$$x_i = 2 \left(\frac{i}{N} + 0.1 \sin \frac{2\pi i}{N} \right) - 1, \quad (144)$$

where $i = 0, 1, 2, \dots, N$, and N is the total number of cell centers in the x -direction. Periodic boundary conditions are imposed at $x = 1$ and $x = -1$; thus, the initial profile of f goes back to exactly the same location at $t = 2$. The red profile in the top panel of Figure 27 shows the initial profile of f at $t = 0$, with the gray vertical lines showing variation of the nonuniform grid geometry. Four distinctive shapes are used in the initial profile: a Gaussian peak, a square wave, a triangle, and a half-circle. This linear advection test simulation has been used extensively in previous studies to test the quality of advection schemes in fluid simulations.

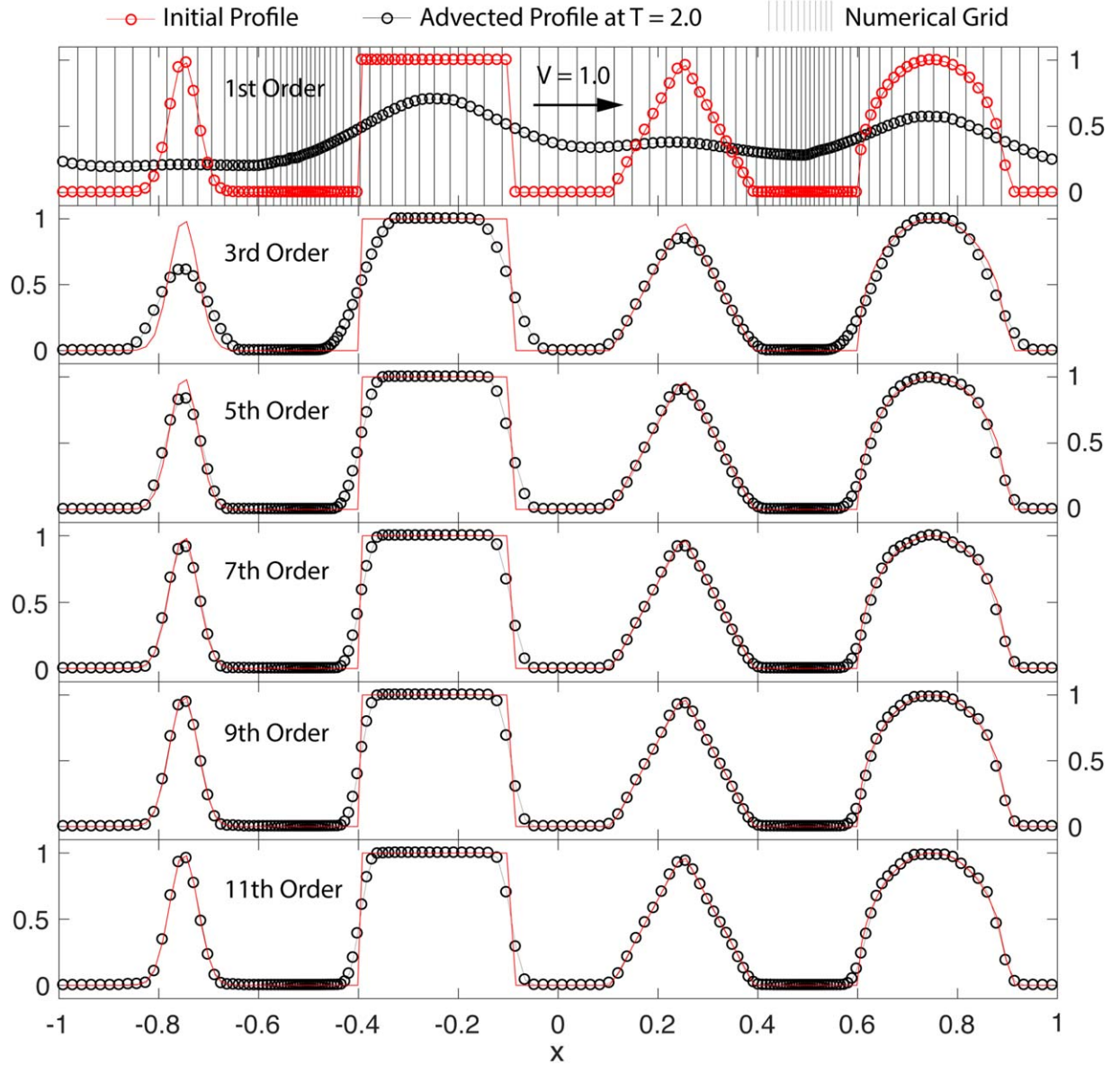


Figure 27. 1D linear advection of four shapes in a nonuniform grid using different orders of reconstruction. The vertical gray line shows the spatial distribution of the computational grid. The red profile is the initial condition at $t = 0$. The black profiles are simulation results at $t = 2.0$ with different orders of spatial reconstruction.

As shown in the initial profile, the smooth resolution is tested by a narrow Gaussian wave with a width of $2.4\Delta x$ centered at $x = -0.75$, while the discontinuity resolution is tested by a square wave centered at $x = -0.25$. Discontinuity in the first derivative is tested using a triangular profile centered at $x = 0.25$. Although there is no discontinuity in the value, the half-circle profile centered at $x = 0.75$ is very challenging since it combines sudden and gradual changes in gradient with a limited number of cells. In addition, there is a discontinuity in curvature at each side of the half-circle profile, which makes it more challenging to resolve the profile especially in nonuniform geometry. In the following simulations, the PDM parameter A described in Appendix C is chosen to be 4.0 and the Courant number is 0.3, which are the same as the default values used in the MHD solver.

The black profiles in Figure 27 are the simulated distributions of f at $t = 2.0$ using six different orders of upwind reconstruction (e.g., first-, third-, ..., 11th-order). The first-order

scheme, which is basically the Donor Cell method, is over-diffusive—all four profiles are smeared significantly as error functions at $t = 2.0$. As the order of reconstruction increases to the seventh order, the algorithm starts to resolve the full narrow Gaussian peak within nine cells. Above the seventh order, the improvement on the Gaussian peak is small. The resolution of the square wave is also improved as the order of reconstruction increases. Above the fifth order, the resolution of the sharp transition of the right side of the square wave remains at four cells. In other words, the improvement on resolving the square wave is small. Given that the total computing time (including both reconstruction and limiting) of the seventh-order scheme is only about 10% more than that of the fifth-order scheme, we choose the seventh-order scheme as the default choice for the spatial reconstruction in the GAMERA code. A more quantitative analysis on the effective numerical diffusion coefficient based on the advection of a square wave is presented in the next section.

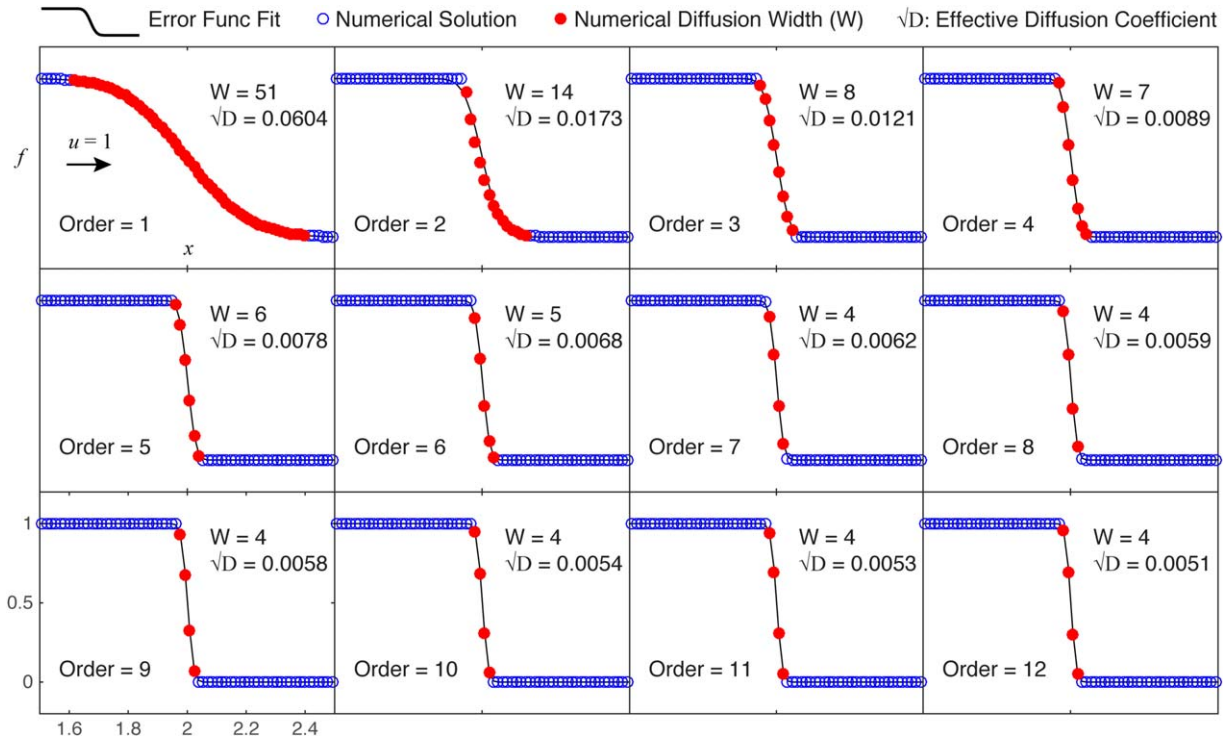


Figure 28. 1D linear advection of a step function using different orders of spatial reconstruction. Blue circles are the numerical solutions to the linear advection equations, while the black curves are least-square fits of analytical solutions to the corresponding numerical solutions. The red dots show the numerical diffusion width, which is defined in the transition region between 0.01 and 0.99. The effective diffusion coefficient \sqrt{D} is listed in the top right corner of each panel.

D.2. The Effective Diffusion Coefficient

When the one-dimensional linear advection Equation (143) is solved numerically, the effective equation solved by the numerical schemes is written as

$$\frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} = D \frac{\partial^2 f}{\partial x^2}, \quad (145)$$

where u is the advection speed and D is an effective diffusion coefficient that depends on both the numerical scheme and the grid resolution. In general, the above equation is a mixed advection–diffusion equation with an analytical solution written as

$$f(x, t) = \int_{-\infty}^{+\infty} f(\xi - ut) e^{-\frac{(\xi-x)^2}{4Dt}} d\xi. \quad (146)$$

When the diffusion coefficient $D \equiv 0$, the solution becomes

$$f(x, t) = f(x - ut, t = 0). \quad (147)$$

We use the numerical solution for a special case of the above linear advection–diffusion equation to approximately quantify the amount of effective numerical diffusion introduced by different reconstruction methods. If the linear advection–diffusion in Equation (145) is solved numerically in a semi-infinite domain $x \in [0, +\infty)$ with an initial condition $f(x, t = 0) = 0$ and a boundary condition $f(x = 0, t) = 1$, the analytical solution in Equation (146) of the advection–diffusion equation with constant u and D is written as

$$f(x, t) = \frac{1}{2} \operatorname{erfc} \left(\frac{x - ut}{\sqrt{Dt}} \right), \quad (148)$$

where $\operatorname{erfc}(\cdot)$ is the complimentary error function. After solving the linear advection Equation (143) numerically with $f(x,$

$t = 0) = 0$ and $f(x = 0, t) = 1$, the numerical diffusion coefficient D is approximated by fitting the numerical solution f_i to the analytical solution (148). To simplify the calculations, we solve the linear advection Equation (143) in a uniform computational domain defined at $[0, 4]$ with 256 cells. At $t = 2$, the front of the step function is located at $x = 2$, and a least-square fit process is applied to the numerical solution f_i to estimate the effective diffusion coefficient D . Figure 28 shows the numerical solutions of the step function using reconstruction schemes from the first order to the 12th order, together with the effective diffusion coefficient \sqrt{D} calculated for each profile. The numerical diffusion width W listed in each panel is defined as the number of computational cells in the transition region of the step function ($0.01 < f_i < 0.99$). If the effective diffusion coefficient D is exactly zero, the numerical diffusion width W is expected to be 0.

The summary of the numerical solutions to the advection Equation (143) is listed in Table 4. Reconstruction schemes from the first order to the 12th order are tested with the analytical solution to the advection–diffusion in Equation (145). The effective diffusion coefficient (\sqrt{D}) is listed in the second column, which decreases from 0.0604 to 0.0051 as the order of reconstruction increases from 1 to 12. The numerical diffusion width (W) is listed in the third column of Table 4. When the order of reconstruction is higher than the seventh order, the numerical diffusion width remains at four, suggesting that the improvement on the slope of the square wave is small above the seventh order. This quantitative comparison is consistent with the linear advection test simulations shown in Figure 27. The third column shows the relative grid Reynolds number in the simulation normalized by the effective grid Reynolds number

Table 4

Numerical Diffusion Coefficient \sqrt{D} , Diffusion Width (W), Relative Reynolds Number R Normalized by the One Derived from the First-order Method, and R/Order Calculated Using Order of Reconstruction from the First Order to the 12th order

Order	\sqrt{D} Fit	Width (cells)	Relative $R \left(\frac{uL}{D} \right)$	R/Order
1	0.0604	51	1.0	1.0
2	0.1730	14	12.1	6.1
3	0.1210	8	24.9	8.3
4	0.0089	7	46.1	11.5
5	0.0078	6	59.9	11.9
6	0.0068	5	78.8	13.1
7	0.0062	4	95.0	13.6
8	0.0059	4	104.8	13.1
9	0.0058	4	108.4	12.0
10	0.0054	4	125.1	12.5
11	0.0053	4	129.8	12.9
12	0.0051	4	140.2	11.6

(uL/D) in the first-order solution. When using a seventh-order reconstruction scheme, the effective grid Reynolds number is enhanced by approximately two orders of magnitude compared to the first-order solution. Above the seventh order, there is still improvement to the relative grid Reynolds number, but the improvement is small. If we use a simple measure (R/Order) to quantify the efficiency of the reconstruction methods, it peaks at the seventh order, as shown in the last column of Table 4.

ORCID iDs

Viacheslav G. Merkin  <https://orcid.org/0000-0003-4344-5424>

References

- Boris, J. P. 1970, A Physically Motivated Solution of the Alfvén Problem, Tech. NRL Memo. Rep. 2167 (Washington, DC: Naval Res. Lab), doi:[10.21236/ad0715774](https://doi.org/10.21236/ad0715774)
- Brambles, O. J., Lotko, W., Zhang, B., et al. 2011, *Sci*, **332**, 1183
- Brecht, S. H., Lyon, J. G., Fedder, J. A., & Hain, K. 1982, *JGR*, **87**, 6098
- Calhoun, D. A., Helzel, C., & LeVeque, R. J. 2008, *SIAMR*, **50**, 723
- Colella, P., Dorr, M., Hittinger, J., & Martin, D. 2011, *JCoPh*, **230**, 2952
- Croisille, J. P., Khanfir, R., & Chanteur, G. 1995, *JSCom*, **10**, 81
- Cunningham, A. J., Frank, A., Varniere, P., Mitran, S., & Jones, T. W. 2011, AstroBEAR: Adaptive Mesh Refinement Code for Ideal Hydrodynamics & Magnetohydrodynamics, Astrophysics Source Code Library, ascl:[1104.002](https://ui.adsabs.org/abs/2011ascl.conf1104C)
- Dobler, W., & Stix, M. 2006, *ApJ*, **638**, 336
- Dubey, A., Reid, L. B., & Fisher, R. 2008, *PhysS*, **132**, 014046
- Evans, C. R., & Hawley, J. F. 1988, *ApJ*, **332**, 659
- Fromang, S., Hennebelle, P., & Teyssier, R. 2006, *A&A*, **457**, 371
- Hain, K. H. 1987, *JCoPh*, **73**, 131
- Huba, J. D. 2003, *LNP*, **615**, 166
- Kallio, E., Luhmann, J. G., & Lyon, J. G. 1998, *JGR*, **103**, 4723
- Leonard, B., & Niknafs, H. 1991, *CF*, **19**, 141
- LeVeque, R. J. 2002, Finite Volume Methods for Hyperbolic Problems, Vol. 31 (Cambridge: Cambridge Univ. Press)
- Luhmann, J. G., Solomon, S. C., Linker, J. A., et al. 2004, *JASTP*, **66**, 1243
- Lyon, J., Fedder, J., & Mobarry, C. 2004, *JASTP*, **66**, 1333
- Lyon, J. G., Brecht, S. H., Huba, J. D., Fedder, J. A., & Palmadesso, P. J. 1981, *PhRvL*, **46**, 1038
- McNutt, R. L., Jr., Lyon, J., & Goodrich, C. C. 1999, *JGR*, **104**, 14803
- Merkin, V. G., & Lyon, J. G. 2010, *JGR*, **115**, A10202
- Merkin, V. G., Lyon, J. G., McGregor, S. L., & Pahud, D. M. 2011, *GeoRL*, **38**, L14107
- Merkin, V. G., Sitnov, M. I., & Lyon, J. G. 2015, *JGRA*, **120**, 1993
- Mignone, A., Bodo, G., Massaglia, S., et al. 2007, *ApJS*, **170**, 228
- Pembroke, A., Toffoletto, F., Sazykin, S., et al. 2012, *JGR*, **117**, A02211
- Powell, K. G., Roe, P. L., Linde, T. J., Gombosi, T. I., & Zeeuw, D. L. D. 1999, *JCoPh*, **154**, 284
- Rilee, M., & Clune, T. 2014, in Proc. 2nd Int. Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, ed. J. C. Carver et al. (Piscataway, NJ: IEEE), 20, doi:[10.1109/SE-HPCSE.2014.5](https://doi.org/10.1109/SE-HPCSE.2014.5)
- Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., & Simon, J. B. 2008, *ApJS*, **178**, 137
- Stone, J. M., & Norman, M. L. 1992, *ApJS*, **80**, 753
- Tóth, G. 1997, in High-Performance Computing and Networking, Vol. 1225, ed. B. Hertzberger & P. Sloot (Berlin: Springer), 253
- Varney, R. H., Wiltberger, M., Zhang, B., Lotko, W., & Lyon, J. 2016, *JGRA*, **121**, 9671
- Wiltberger, M., Wang, W., Burns, A., et al. 2004, *JASTP*, **66**, 1411
- Xu, K. 1999, *JCoPh*, **153**, 334
- Yee, K. 1966, *ITAP*, **14**, 302
- Zalesak, S. T. 1979, *JCoPh*, **31**, 335
- Zhang, B., Delamere, P. A., Ma, X., et al. 2018, *GeoRL*, **45**, 56
- Zhang, B., Sorathia, K. A., Lyon, J. G., et al. 2019a, Supplementary Material for GAMERA: A Three-dimensional Finite-volume MHD Solver for Non-orthogonal Curvilinear Geometries, v1, Zenodo, doi:[10.5281/zenodo.3364572](https://doi.org/10.5281/zenodo.3364572)
- Zhang, B., Sorathia, K. A., Lyon, J. G., Merkin, V. G., & Wiltberger, M. 2019b, *JCoPh*, **376**, 276
- Ziegler, U. 2008, *CoPhC*, **179**, 227
- Zilhao, M., & Noble, S. C. 2014, *CQGra*, **31**, 065013