Parallel Adaptive Solution of the MHD Equations and Its Role in the Space-Weather Modeling Framework

Kenneth G. Powell, Tamas I. Gombosi, Quentin F. Stout, Darren L. De Zeeuw, Gábor Tóth, Igor V. Sokolov, Aaron J. Ridley, Kenneth C. Hansen, Ward B. Manchester, and Ilia I. Roussev

Center for Space Environment Modeling, The University of Michigan, Ann Arbor Michigan

Abstract. Over the last ten years, a collaboration of researchers in space physics, computer science and computational fluid dynamics has grown into the Center for Space Environment Modeling at the University of Michigan. The group has partnered with researchers at Rice, Stanford, NCAR, and NASA Goddard, among other institutions, to develop a first-principles-based software framework for modeling space-weather events.

In this paper, a solution-adaptive method for solving the MHD equations in a highly efficient manner on parallel computers is presented. In addition, an overview of the broader effort to develop the Space Weather Modeling Framework is given.

1. Introduction

Space weather has been used to refer to the conditions on the Sun and in the solar wind, magnetosphere, ionosphere, and thermosphere that can influence the performance and reliability of space-borne and ground-based technological systems or can endanger human life or health. The solar corona is so hot (> 10⁶ K) that in open magnetic field regions it undergoes a transonic expansion, filling all of interplanetary space with a supersonic magnetized plasma flowing radially outward from the Sun. As this flowing plasma, which is called the solar wind, passes the Earth, it interacts strongly with the geomagnetic field, severely compressing the field on the dayside of the Earth, and drawing it out into a long, comet-like tail on the nightside. The confined region of geomagnetic field is called the Earth’s magnetosphere. Significant temporal variations of solar wind speed at the orbit of Earth occur due to the rotation of solar wind structures. Such variations can also be produced by the transient ejection of mass and magnetic field from the solar corona (coronal mass ejections or CMEs). Indeed, the most severe storms experienced in the Earth’s space environment are driven by exceptionally fast CMEs that exhibit a strong southward magnetic field component throughout a significant fraction of their volume. These very fast CMEs, which are ejected from the corona at speeds of more than 1000 km/s, also drive strong hydromagnetic shocks. These shocks are efficient producers of energetic particles. Of course, a very fast CME is only effective in producing a severe geomagnetic storm when it is directed towards the Earth, and this fact presents a problem for those attempting to give forewarning of such storms.
The solar wind not only confines the terrestrial magnetic field within the magnetospheric cavity, but it also transfers significant mass, momentum, and energy to the magnetosphere, as well as to the ionosphere and upper atmosphere. One dramatic consequence of this interaction between the solar wind and the magnetosphere is the production of a variety of complex electric current systems, ranging from a sheet of current flowing on the boundary between the solar wind and magnetosphere, to an enormous ring of current flowing around the Earth in the inner magnetosphere, to currents flowing throughout the ionosphere and connecting along magnetic field lines to magnetospheric currents systems. Another result of the solar-wind/magnetosphere interaction is the production of highly energetic particles that are stored in the magnetosphere and precipitated into the upper atmosphere. Both the electric currents and the energetic particles can have severe consequences for a number of human activities, all the way from the ground to space. It is the variation over time of these electric current systems and energetic particle populations in the geospace environment that modulates the consequences for human activities, and that is consequently the source of what we refer to as space weather. A CME and its interaction with the magnetosphere is illustrated in Figure 1. The magnetic cloud generated by the CME approaches the magnetosphere in the top frame. In the bottom frame the cloud interacts with the magnetosphere and generates stronger magnetospheric current systems and larger, more energetic magnetospheric particle populations, a phenomenon which is called a geomagnetic storm. During magnetic storms the magnetospheric topology is significantly modified and large transients are generated. As solar activity increases, the frequency of CMEs is substantially increased, and the severity of space weather is concomitantly increased.

2. A Core Solver Piece — Adaptive, Parallel, Ideal MHD

The solar wind can, in much of the region between Earth and the Sun, be modeled as an ideal, non-relativistic, compressible plasma. The governing equations
for an ideal, non-relativistic, compressible plasma may be written in a number of different forms. While the different forms of the MHD equations describe the same physics at the differential equation level, there are important practical differences when one solves discretized forms of the various formulations.

2.1. Fully Conservative Form

The fully conservative form of the equations is

\[
\frac{\partial U}{\partial t} + (\nabla \cdot \mathbf{F})^T = 0, \tag{1}
\]

where \( U \) is the vector of conserved quantities and \( \mathbf{F} \) is a flux diad,

\[
U = \begin{pmatrix}
\rho \\
\rho u \\
B \\
E_{\text{mhd}}
\end{pmatrix}, \tag{2}
\]

\[
\mathbf{F} = \begin{pmatrix}
\rho u \\
\rho uu + \left(p + \frac{1}{2\mu_0}B^2\right)\mathbf{I} - \frac{1}{\mu_0}BB \\
u\left(E_{\text{mhd}} + p + \frac{1}{2\mu_0}B^2\right) - \frac{1}{\mu_0}(u \cdot B)B
\end{pmatrix}^T, \tag{3}
\]

where \( E_{\text{mhd}} \) is the magnetohydrodynamic energy, given by

\[
E_{\text{mhd}} = \frac{1}{2} \rho u^2 + \frac{1}{\gamma - 1} p + \frac{1}{2\mu_0}B^2. \tag{4}
\]

2.2. Symmetrizable Formulation

Symmetrizable systems of conservation laws have been studied by a number of authors, including Godunov (1972) and Harten (1983). One property of the symmetrizable form of a system of conservation laws is that an added conservation law

\[
\frac{\partial (\rho s)}{\partial t} + \frac{\partial (\rho s u_x)}{\partial x} + \frac{\partial (\rho s u_y)}{\partial y} + \frac{\partial (\rho s u_z)}{\partial z} = 0
\]

for the entropy \( s \) can be derived by a linear combination of the system of equations. For the ideal MHD equations, as for the gasdynamic equations, the entropy is \( s = \log(p/\rho^\gamma) \). Another property of symmetrizable systems is that they are Galilean invariant; all waves in the system propagate at speeds \( u \pm c_w \) (for MHD, the possible values of \( c_w \) are the Alfvén, magnetofast and magnetoslow speeds). Neither of these properties holds for the fully conservative form of the MHD equations.

Godunov showed that the fully conservative form of the MHD equations (1) is not symmetrizable (Godunov 1972). The symmetrizable form may be written as

\[
\frac{\partial U}{\partial t} + (\nabla \cdot \mathbf{F})^T = \mathbf{Q}, \tag{5}
\]
where

\[ Q = -\nabla \cdot B \begin{pmatrix} 0 \\ \frac{1}{\mu_0} B \\ \frac{1}{\mu_0} u \cdot B \end{pmatrix} . \] (6)

Powell (1994) showed that this symmetrizable form can be used to derive a Roe-type approximate Riemann solver for solving the MHD equations in multiple dimensions.

The MHD eigensystem arising from Eq. (1) or Eq. (5) leads to eight eigenvalue/eigenvector pairs. The eigenvalues and associated eigenvectors correspond to an entropy wave, two Alfvén waves, two magnetofast waves, two magnetoslow waves, and an eighth eigenvalue/eigenvector pair that depends on which form of the equations is being solved. This last wave (which describes the jump in the normal component of the magnetic field at discontinuities) has a zero eigenvalue in the fully conservative case, and an eigenvalue equal to the normal component of the velocity, \( u_n \), in the symmetrizable case. The expressions for the eigenvectors, and the scaling of the eigenvectors, are more intricate than in gasdynamics (Roe and Balsara 1996).

The symmetrizable formulation (given by Eq. 5) is formally not fully conservative. Terms of order \( \nabla \cdot B \) are added to what would otherwise be a divergence form. The danger of this is that shock jump conditions may not be correctly met, unless the added terms are small, and/or they alternate in sign in such a way that the errors are local, and in a global sense cancel in some way with neighboring terms. This downside, however, has to be weighed against the alternative; a system (i.e., the one without the source term) that, while conservative, is not Galilean invariant, has a zero eigenvalue in the Jacobian matrix, and is not symmetrizable. In practice, using the symmetrizable formulation, and adding a technique to keep the magnitude of \( \nabla \cdot B \) small, is a viable approach.

2.3. The Upwind Finite-Volume Scheme

The MHD equations are well suited for finite-volume methods when the governing equations are integrated over a computational cell \( i \), yielding

\[ \frac{dU_i}{dt} = -\frac{1}{V_i} \sum_{\text{faces}} F \cdot \hat{n} A - \frac{Q_i}{V_i} \sum_{\text{faces}} B \cdot \hat{n} A , \] (7)

where \( V_i \) is the volume of cell \( i \), \( A \) is the surface area of the faces forming the computational cell, \( \hat{n} \) is the unit vector normal to the cell faces, \( U_i \) is the cell-averaged conserved solution vector, and \( Q_i \) is given by

\[ Q_i = -\begin{bmatrix} 0 \\ \frac{1}{\mu_0} B_i \\ \frac{1}{\mu_0} u_i \cdot B_i \end{bmatrix} . \] (8)

The numerical face fluxes, \( F \cdot \hat{n} \), are defined in terms of the left and right interface solution states, \( U_L \) and \( U_R \), as follows

\[ F \cdot \hat{n} = \mathcal{F} (U_L, U_R, \hat{n}) , \] (9)
where $\mathbf{U}_L$ and $\mathbf{U}_R$ are the state vectors at the left and right sides of the interface.

Because the MHD equations are a system of hyperbolic conservation laws, many of the techniques that have been developed for the Euler equations can be applied relatively straightforwardly. In particular, the high-resolution finite-volume approach of van Leer (1979) (i.e. approximate Riemann solver + limited interpolation scheme + multi-stage time-stepping scheme) is perfectly valid. The Rusanov/Lax–Friedrichs approximate Riemann solver can be applied directly; no knowledge of the eigensystem of the MHD equations is required other than the fastest wave speed in the system. A Roe-type scheme (Roe 1981) can be constructed for non-relativistic MHD (Powell et al. 1999), but requires more work, because of the complexity of the eigensystem. In addition, an HLLE-type Riemann solver has been derived by Linde (1998); it is less dissipative than the Rusanov/Lax–Friedrichs scheme, but less computationally intensive than the Roe scheme. Whichever approximate Riemann solver is chosen to serve as the flux function, standard interpolation schemes and limiters can be used to construct a finite-volume scheme.

One way in which the numerical solution of the MHD equations differs from that of the gasdynamic equations is the constraint that $\nabla \cdot \mathbf{B} = 0$. Enforcing this constraint numerically, particularly in shock-capturing codes, can be done in a number of ways, but each way has its particular strengths and weaknesses. Tóth (2000) has published a numerical comparison of many of the approaches for a suite of test cases. Several of the approaches in that paper have been implemented in the adaptive, parallel MHD code described here. In practice, the Powell–Roe and HLLE schemes mentioned above are most commonly used by users of the code. The projection scheme described by Tóth (2000) is used to keep $\nabla \cdot \mathbf{B}$ small, with the frequency of its use depending on the problem being solved.

2.4. Block-Based AMR for MHD

For typical solar-wind flows, length scales can range from tens of kilometers in the near-Earth region to the Earth-Sun distance ($1.5 \times 10^{11}$ m), and timescales can range from a few seconds near the Sun to the expansion time of the solar wind from the Sun to the Earth ($\sim 10^5$ s). The use of AMR is not only extremely beneficial, but a virtual necessity for solving problems with such disparate spatial and temporal scales.

Building on prior work by Berger & Saltzman (1994) and Quirk & Hanebutte (1993), and keeping in mind the desire for high performance on massively parallel computer architectures, a relatively simple yet effective block-based AMR technique has been developed and is used in conjunction with the finite-volume scheme described above. Here the governing equations are integrated to obtain volume-averaged solution quantities within rectangular Cartesian computational cells. The computational cells are embedded in regular structured blocks of equal sized cells. The blocks are geometrically self-similar with dimensions $\tilde{\ell}_x \times \tilde{\ell}_y \times \tilde{\ell}_z$ and consist of $N_x \times N_y \times N_z$ cells, where $\tilde{\ell}_x$, $\tilde{\ell}_y$, and $\tilde{\ell}_z$ are the nondimensional lengths of the sides of the rectangular blocks and $N_x$, $N_y$, and $N_z$ are even, but not necessarily all equal, integers. Typically, blocks consisting of anywhere between $4 \times 4 \times 4 = 64$ and $12 \times 12 \times 12 = 1728$ cells are used (see Figure 2). Solution data associated with each block are stored in standard
indexed array data structures. It is therefore straightforward to obtain solution information from neighboring cells within a block.

Computational grids are composed of many self-similar blocks. Although each block within a grid has the same data storage requirements, blocks may be of different sizes in terms of the volume of physical space that they occupy. Starting with an initial mesh consisting of blocks of equal size (i.e., equal resolution), adaptation is accomplished by the dividing and coarsening of appropriate solution blocks. In regions requiring increased cell resolution, a “parent” block is refined by dividing itself into eight “children” or “offspring.” Each of the eight octants of a parent block becomes a new block having the same number of cells as the parent and thereby doubling the cell resolution in the region of interest. Conversely, in regions that are deemed overresolved, the refinement process is reversed, and eight children are coarsened and coalesced into a single parent block. In this way, the cell resolution is reduced by a factor of 2. Standard multigrid-type restriction and prolongation operators are used to evaluate the solution on all blocks created by the coarsening and division processes, respectively.

The decision of where to adapt is made based on user-determined adaptation criteria. The user can choose up to three criteria from a large list (typical criteria are $|\nabla \times B|$, $|\nabla \times u|$, $|\nabla p|$). Each block is assigned a value equal to the maximum over all cells of the quantity being used in the adaptation criterion. The blocks are then sorted by these values. The top of the list is flagged for refinement; the bottom of the list is flagged for coarsening. The number of blocks flagged for refinement and coarsening is based on user-specified rules for the percentage of blocks to be refined or coarsened. Two neighboring blocks, one of which has been refined and one of which has not, are shown in Figure 2. Any of the blocks shown in Figure 2 can in turn be refined, and so on, leading to successively finer blocks. In the present method, mesh refinement is constrained such that the cell resolution changes by only a factor of 2 between adjacent blocks and such that the minimum resolution is not less than that of the initial mesh.
In order that the update scheme for a given iteration or time step can be applied directly to all blocks in an independent manner, some additional solution information is shared between adjacent blocks having common interfaces. This information is stored in an additional two layers of overlapping “ghost” cells associated with each block as shown in Figure 2. At interfaces between blocks of equal resolution, these ghost cells are simply assigned the solution values associated with the appropriate interior cells of the adjacent blocks. At resolution changes, restriction and prolongation operators, similar to those used in block coarsening and division, are employed to evaluate the ghost cell solution values. After each stage of the multistage time-stepping algorithm, ghost cell values are reevaluated to reflect the updated solution values of neighboring blocks. With the AMR approach, additional interblock communication is also required at interfaces with resolution changes to strictly enforce the flux conservation properties of the finite-volume scheme. In particular, the interface fluxes computed on more refined blocks are used to correct the interface fluxes computed on coarser neighboring blocks so as to ensure that the fluxes are conserved across block interfaces.

2.5. Explicit Time-Stepping

The parallel block-based AMR solver was designed from the ground up with a view to achieving very high performance on massively parallel architectures. The underlying upwind finite-volume solution algorithm, with explicit time stepping, has a very compact stencil and is therefore highly local in nature. The hierarchical data structure and self-similar blocks make domain decomposition of the problem almost trivial and readily enable good load-balancing, a crucial element for truly scalable computing. A natural load balancing is accomplished by simply distributing the blocks equally among the processors. Additional optimization is achieved by ordering the blocks using the Peano–Hilbert space filling curve to minimize inter-processor communication. The self-similar nature of the solution blocks also means that serial performance enhancements apply to all blocks and that fine-grain parallelization of the algorithm is possible. The parallel implementation of the algorithm has been carried out to such an extent that even the grid adaptation is performed in parallel.

Other features of the parallel implementation include the use of FORTRAN 90 as the programming language and the message-passing interface (MPI) library for performing the interprocessor communication. Use of these standards greatly enhances the portability of the code and leads to very good serial and parallel performance. The message passing is performed in an asynchronous fashion with gathered wait states and message consolidation.

Implementation of the algorithm has been carried out on Cray T3E supercomputers, SGI and Sun workstations, on Beowulf type PC clusters, on SGI shared-memory machines, on a Cray T3D, and on several IBM SP2s. The code scales nearly perfectly to 1,500 processors and a sustained speed of 342 GFlops has been attained on a Cray T3E-1200 using 1,490 PEs. For each target architecture, simple single-processor measurements are used to set the size of the adaptive blocks. The scaling on various architectures is shown in Figure 3.
2.6. Implicit Time-Stepping

A number of time-stepping algorithms have been implemented in the AMR MHD solver. The simplest and least expensive scheme is a multistage explicit time stepping, for which the time step is limited by the CFL stability condition. An unconditionally stable fully implicit time stepping scheme (Tóth et al. 1998) has also been implemented. The second-order implicit time discretization (BDF2) (Bank 1985) requires the solution of a non-linear system of equations for all the flow variables. This can be achieved by the Newton–Krylov–Schwarz approach: a Newton iteration is applied to the non-linear equations; a parallel Krylov type iterative scheme is used to solve the linear systems; the convergence of the Krylov solver is accelerated with a Schwarz type preconditioning. Two Krylov solvers have been implemented: BiCGSTAB and GMRES. A modified block incomplete LU (MBILU) preconditioner is applied on a block by block basis. Since every block has a simple Cartesian geometry, the preconditioner can be implemented very efficiently. The resulting implicit scheme requires about 20–30 times more CPU time per time step than the explicit method, but the physical time step can be 1,000 to 10,000 times larger. This implicit algorithm has a very good parallel scaling due to the Krylov scheme and the block by block application of the preconditioner.

In addition, it is possible to combine explicit and implicit time stepping. Magnetosphere simulations include large volumes where the Alfvén speed is quite low (tens of km/s) and the local CFL number allows large explicit time steps (tens of seconds to several minutes). In these regions implicit time stepping is a waste of computational resources. Since the parallel implicit technique is fundamentally block based, only those blocks where the CFL condition would limit the explicit time step to less than the selected time step (typically ∼
10 s) are treated implicitly. Needless to say, this combined explicit-implicit time stepping represents more computational challenges (such as separate load balancing of explicit and implicit blocks). This approach is described more fully by Tóth et al. (2006).


In a number of fields in which computer-based modeling of complex, multi-scale, multi-physics problems plays an important role, software frameworks are being developed. The term software framework lacks a unique definition: some groups refer to a collection of models that interact through a coupling mechanism, however simple or intricate the coupling, as a software framework; other groups refer to the coupling software itself, independent of the models that it can couple, as a software framework. The Space-Weather Modeling Framework (SWMF) falls under the coupling-plus-models definition of a framework. It aims at providing a flexible and extensible software architecture for multi-component physics-based space weather simulations, as well as for various space physics applications. The main design goals are:

- incorporate computational physics modules with only modest modification;
- achieve good parallel performance in the coupling of the physics components; and
- allow physics components to interact within the SWMF as efficiently as possible.

The SWMF includes a superstructure layer that drives the coupled-model application, and an infrastructure layer that provides utilities and data structures for model developers. For each model a wrapper needs to be written that can make use of the infrastructure layer. The individual models are coupled together with the efficient SWMF parallel coupling toolkit. The SWMF is a fully functional and documented framework that provides a high-performance computational capability to simulate the physics from the low solar corona to the upper atmosphere of the Earth. The SWMF is described more fully in several articles (Tóth et al. 2005) and the SWMF web site\(^1\). One of the most important features of the SWMF is that it can incorporate different computational physics modules to model different domains of the Sun-Earth system. Each module for a particular domain can be replaced with alternatives, and one can use only a subset of the modules if desired. The modules that currently comprise SWMF are:

- Solar Corona (SC)
- Eruptive Event Generator (EE)

\(^1\)http://csem.engin.umich.edu/SWMF
Figure 4. Depiction of modules in SWMF

- Inner Heliosphere (IH)
- Solar Energetic Particles (SP)
- Global Magnetosphere (GM)
- Inner Magnetosphere (IM)
- Radiation Belt (RB)
- Ionosphere Electrodynamics (IE)
- Upper Atmosphere (UA).

The modules are represented pictorially in Figure 4. The modules are coupled together by the framework, through a control module. The control module determines the overall time-stepping of the code, the parallel decomposition of the models, the initiation and termination of the model runs, and the saving of restart files of the models. This involves code that determines when the coupling should occur, how it happens, grid interpolation, message passing between different components, and synchronization of the model runs to allow for a physically meaningful coupling. The SWMF uses a component architecture (Figure 5), with each component created from a physics module by making some minimal changes and by adding two relatively small units of code:
Figure 5. (left) The architecture of the SWMF and (right) the structure of a physics module in it.

- a wrapper, which provides the standard interface to control the physics module;
- and a coupling interface, to perform the data exchange with other components.

Both the wrapper and the coupling interface are constructed from building blocks provided by the framework. The structure of a component and its interaction with the Control Module (CON) and another component are illustrated in Figure 5.

4. Representative Results of the Coupled Model

A real test of the SWMF and its coupled models is the calculation of an energetic CME, its propagation, and its interaction with the Earth’s magnetosphere. One of the most energetic CMEs observed was the October 28, 2003 event, commonly referred to as the Halloween storm. The Solar Corona and Inner Heliosphere modules provided an initial, rotating solution consistent with solar-surface observations from the time. These modules, based on the parallel adaptive MHD solver described above, were run with 2.5 million cells, as small as $3 \times 10^{-3} R_{\odot}$, in the Solar Corona, and more than 16 million cells, ranging from $0.25 R_{\odot}$ to $4 R_{\odot}$, in the Inner Heliosphere region (where $R_{\odot}$ is the solar radius). The Eruptive Event Generator was used to initiate a CME, with the location, orientation and strength set to match observations. As the solution was advanced in time, the CME grew and propagated towards Earth.
The interaction of the CME with Earth’s magnetosphere and upper atmosphere were calculated by the Global Magnetosphere, Inner Magnetosphere, Ionosphere Electrodynamics and Upper Atmosphere models. Figure 6 shows a representation of the magnetosphere at 08 UT on October 29, 2003, as computed by SWMF. The simulation captured the large ring current and the high pressure associated with it, thanks to the drift physics model in the Inner Magnetosphere module.

Comparisons between simulation results and satellite observations are shown in Figure 7. The three satellites are: Cluster, which was in the dayside magnetosphere; Polar, which was near the northern cusp; and GOES-10, which was behind the Earth at geosynchronous orbit. The results show that the simulation reproduces the observations in the dayside magnetosphere and in the cusp region extremely well. The agreement with Cluster and Polar is very good. In the closed field line region behind Earth the agreement is reasonable but there are several important features that are missed by the simulation. On the positive side, the simulation reproduces the Bz component, and thus captures the motion of the magnetopause. However, the transient feature around 2300 UT is missed by the simulation.
Figure 7. Comparison of the magnetic field measured by the Cluster, Polar and GOES-10 satellites (blue) with simulation results (black) for the October 29-30, 2003 storm.
5. Concluding Remarks

The ability to model and predict space weather is extremely important in protecting satellites, astronauts, and even ground-based systems that are affected by upper atmosphere disturbances. The complex multiscale, multiphysics nature of the problem make development of a software framework for these simulations a multi-person, multi-year effort. The progress made on the Space-Weather Modeling Framework, described above, represents a good start towards the overarching goal of first-principles-based modeling of space weather.

Acknowledgments. This work has been supported by NASA AISRP grant NNG04GP89G, NASA ESS CT cooperative agreement NCC5-614, and by DoDMURI grant F49620-01-1-0359. We gratefully acknowledge the supercomputing resources provided by NASAs Columbia system under award SMD1-Dec04-0099.

References