Cosmos++: Relativistic Magnetohydrodynamics on Unstructured Grids with Local Adaptive Refinement

Jay D. Salmonson, Peter Anninos, P. Chris Fragile(2), and Karen Camarda(3)  
Lawrence Livermore National Laboratory, Livermore, CA 94551  
(2) College of Charleston  
(3) Washburn University  
E-mail: salmonson@llnl.gov

Abstract. A code and methodology are introduced for solving the fully general relativistic magnetohydrodynamic (GRMHD) equations using time-explicit, finite-volume discretization. The code has options for solving the GRMHD equations using traditional artificial-viscosity (AV) or non-oscillatory central difference (NOCD) methods, or a new extended AV (eAV) scheme using artificial-viscosity together with a dual energy-flux-conserving formulation. The dual energy approach allows for accurate modeling of highly relativistic flows at boost factors well beyond what has been achieved to date by standard artificial viscosity methods. It provides the benefit of Godunov methods in capturing high Lorentz boosted flows but without complicated Riemann solvers, and the advantages of traditional artificial viscosity methods in their speed and flexibility. Additionally, the GRMHD equations are solved on an unstructured grid that supports local adaptive mesh refinement using a fully threaded oct-tree (in three dimensions) network to traverse the grid hierarchy across levels and immediate neighbors. Some recent studies will be summarized.

1. Introduction  
Cosmos++ is a massively parallel, multi-dimensional astrophysics code which solves the general relativistic magnetohydrodynamic (GRMHD) equations on an unstructured, adaptive mesh using time-explicit, finite-volume discretization. The flexibility of the modular, object-oriented design of Cosmos++ allows for the implementation of numerous solution methods of the GRMHD equations to a wide variety of problems as well as the addition of new physics packages.

2. The Code  
Cosmos++ [1] is a massively parallel, multi-dimensional, fully covariant, modern object-oriented (C++) magnetohydrodynamics code written to support structured and unstructured adaptively refined meshes, and for both Newtonian and general relativistic astrophysical applications. It includes numerous hydrodynamics solvers (conservative and non-conservative), magnetic fields (ideal and non-ideal), radiation multi-group flux-limited diffusion, a network of more than 30 chemical gas-phase reactions, relativistic scalar (inflaton) fields for the early universe, isotropic FRW cosmologies, radiative cooling, dark matter particles, self-gravity, geodesic transport, and
generic tracer fields. In general relativistic mode, an evolving metric, determined by solving
Einstein’s equation, is currently under development (§3.3).

Cosmos++ is actively developed by numerous scientists and has been ported to more than
da dozen platforms including stand alone Linux workstations. We use the Perforce control
system to coordinate code development. The Message Passing Interface (MPI), with non-
blocking communications, is used for parallel decomposition, providing a portable mechanism
for communication of data between partitioned domains.

The adaptive mesh refinement (AMR) framework differs from the more standard approach
by refining individual cells rather than introducing patches of sub-grids composed of multi-
dimensional arrays of cells, thus providing greater flexibility in modeling complex flows and
greater efficiency in positioning computational resources. All cells in the mesh are fully threaded
with local binary, quad, or oct-tree parent-child hierarchies with fast and efficient direct neighbor
access. Due to the covariant nature of the mathematical formulations of the grid geometries
incorporated into Cosmos++, the mesh can be adaptively refined even if the zones are allowed to
move around in physical space, providing even greater computational efficiency over traditional
methods for tracking and resolving flow features. Cosmos++ is, in effect, an AMR scheme
activated in a covariant Arbitrary-Lagrange-Eulerian (ALE) computational framework.

Cosmos++ is designed using object-oriented principles to create abstraction classes for vector
and tensor (both three and four dimensional) functions on which most mathematical operations
are based. It takes advantage of the operator overload, inheritance, polymorphism and virtual
methods features of the C++ language in its design of the zone, mesh, refinement, physics,
boundary condition, data exchange, and output classes. This simplifies the front-end user
interface considerably, provides user interactivity with the simulation as it runs, minimizes
code replication, and allows for the physics packages and supporting framework to be easily
developed and expanded. All data structures (mesh and fields) are stored as vector or map
container classes using the C++ Standard Template Library (STL) which provides convenient
memory access functions and other useful support features (e.g., sorting and searching) for
interacting with the data.

Numerical and mathematical operations are performed in the physics packages only across
the list of leaf zones in a grid hierarchy which have no children, so the effective mesh presented to
the user or developer is thus unstructured in general, and composed of many different arbitrarily
shaped zones with unconstrained nodal connectivity. However, from a developer’s point of view,
the interface to this complicated mesh structure, at any discrete instant in time, appears quite
simply as a single-level mesh with simple linked list iterators that can be used to access each zone
(and its neighbors) in succession. The physics packages are thus easily interchangeable with any
other C++ code (single or multi-level) which accommodates encapsulated class objects. The
portability and interchangeability of physics classes and modules is something we have stressed
in the design of Cosmos++.

3. Recent Results

3.1. Relativistic Blastwaves and Afterglows

A couple of the hydrodynamic packages of Cosmos++ evolve total energy instead of (or, in a
hybrid dual mode, in addition to) internal energy. This allows high velocity, relativistic flows
and shocks to be accurately modeled. Also, adaptive mesh refinement is crucial to resolve the
nearby, length contracted relativistic blastwaves. Finally, Cosmos++ has sophisticated,
flexible post-processing whereby the simulation’s stepped time slicing can be transformed to an
observer’s time slicing. We recently used these components to study the December 17, 2004
giant flare of Soft Gamma-ray Repeater 1806-20 [2]. In that study 2D axisymmetric simulations
of mildly relativistic ($\gamma \sim 1.1 - 1.7$) ejecta colliding with an interstellar medium were performed
(Fig. 1) and then post-processed for a range of inclinations of the jet axis with respect to the
observer. We were able to quantitatively compare, contrast and constrain two models for the ejecta collision. One key result of this work was the demonstration that the ejecta had to be structured; i.e. more rapid expansion in the ejecta core than in the wings was required to explain the observations. We thus identified key places where the analytical models for jetted blastwaves had failed.

3.2. Tilted Disks
Global magnetohydrodynamic simulations of tilted thick-disk accretion have recently been performed with Cosmos++[3]. In this study was performed the first 3D fully relativistic numerical simulations of tilted accretion disks around Kerr black holes. These demanding calculations required hundreds of processors and were evolved for millions of timesteps. Toward this goal, significant improvements in scaling were achieved (Fig. 2). In these simulations a seed magnetic field is amplified via the Magneto-Rotational Instability to become dynamically significant. This field thereby enables and mediates mass accretion onto the blackhole. We discovered a rich, complex dynamic between the inner disk and the spinning black hole whereby material is grabbed from the disk as it interacts with the hole’s innermost stable circular orbit (ISCO) and forms dual, opposing plunging streams which fall into the hole while undergoing Lense-Thirring precession (Fig. 3). The dynamics of these hot plunging streams have direct ramifications for observations, which are currently being explored.

Figure 1. Sequence of snapshots of material ejected at velocity of 0.5c from Soft Gamma-ray Repeater 1806-20 plowing into the interstellar medium. Matter density is displayed on the right and energy density on the left. The lower right frame shows the adaptive mesh.
Figure 2. Plot of the total compute time as a function of the CPU count for a series of test runs carried out on the NCSA Teragrid cluster (Mercury). Data is normalized to the compute time of the CPU test. The horizontal line represents ideal linear scaling.

Figure 3. Dual, opposing streams are stripped from the inner tilted accretion disk and plunge into the black hole.
3.3. Einstein Solver

A significant augmentation to the physics capabilities of Cosmos++ is the recent addition of an Einstein solver. This allows the space-time metric to be evolved dynamically according to Einstein’s general relativity. The code’s generalized formulation of the magnetohydrodynamic equations with an arbitrary metric significantly facilitates this addition. The Einstein equations are solved with a 3+1 space-time split. Testing is currently underway, including the propagation of linear vacuum space-time Minkowski waves (Fig. 4).

4. Considerations for the Petascale

The flexible, modular design of Cosmos++ and its use of proven, extendable hydrodynamic algorithms makes it an excellent framework for developing and testing new physics and new numerical methods. For example, current efforts include research of novel radiation algorithms. Being a code based on an unstructured mesh, one of the biggest efficiency challenges for Cosmos++ is irregular access to data in memory. Mesh neighbor cells cannot be expected to be neighbors in memory. Thus Cosmos++ does not benefit from optimizations based on proximate data residing in cache. A key direction in future efforts to optimize the scaling of Cosmos++ to the petascale will be to optimally order data access and so limit memory latency and improve cache coherency. These issues are nearly universal to modern, flexible scientific codes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] Anninos, P., Fragile, P. C., & Salmonson, J. D. 2005, The Astrophysical Journal, 635, 723
[2] Salmonson, J. D., Fragile, P. C., & Anninos, P. 2006, The Astrophysical Journal, 652, 1508
[3] Fragile, P. C., Blaes, O., Anninos, P., Salmonson, J. D. 2007, to appear in The Astrophysical Journal (astro-ph/0706.4303)