User Guide Part 1: Introduction

Author: John Hayes

The Short Story

ZEUS-MP is the latest addition to the ZEUS line of community application codes developed by the Laboratory for Computational Astrophysics. The "MP" suffix denotes the "multi-physics," "massively parallel," and "message passing" aspects of the code. The physics suite in this release of ZEUS-MP includes gas hydrodynamics, ideal MHD, implicit flux-limited radiation diffusion (FLD), self gravity, and multispecies advection. The HD, MHD, and FLD modules can be used on 1, 2, or 3-dimensional grids, in Cartesian, cylindrical, or spherical coordinate geometries. Likewise, self-gravity is supported in all dimensions and can be calculated with a variety of methods, depending on geometry, dimensionality, and boundary conditions. Methods include: (1) GM/r on 1D or 2D spherical grids; (2) a conjugate-gradient Poisson solver on 2D or 3D spherical and cylindrical grids; (3) a multigrid-based Poisson solver on 3D cartesian grids with non-periodic boundary conditions, and (4) a Fast Fourier Transform solver for 3D cartesian grids with triply-periodic boundaries. External point-mass potentials are also supported.

Multiple Versions of ZEUS-MP

There now exist at least 3 different versions of ZEUS-MP which have been made publicly available. The first of these was released directly by the Laboratory for Computational Astrophysics in 1999 when the LCA was part of the National Center for Supercomputing Applications.

A modified version of the original 1.0b code, currently (12/01/05) dubbed version 1.5.8a, has been distributed by John Vernaleo and Chris Reynolds on their web site. Additional information about the code and a paper recently posted to Astro-ph (Vernaleo and Reynolds 2005; astro-ph/0511501) is available there.

The version of ZEUS-MP distributed on this page, which I am currently calling version 2.0, shares a common ancestor (v1.0b) with Vernaleo's ZEUS-MP v1.5.8a, but similarities end there. In what follows, **NO COMMENTS MADE ABOUT V1.0B OR VERSION 2 SHOULD BE ASSUMED TO APPLY TO V1.5.8A**.

The following discussion compares ZEUS-MP v2.0, distributed on this web site, to ZEUS-MP v1.0b, the first public release of ZEUS-MP. Differences between the two fall into three categories:

1. **Available Physics:** Flux-limited radiation diffusion (FLD) and multispecies fluid advection are new additions to the ZEUS-MP physics set.
2. **Application Flexibility:** Both the HD and MHD modules in v1.0b were hard-wired for 3-D execution. In version 2, HD, MHD, and RHD are possible in all dimensions on cartesian (X-Y-Z), cylindrical (Z-R-Phi), and spherical (R-Theta-Phi) grids. In addition, non-ideal equations of state are now supported, though none are included in this release.
3. **Software Engineering:** In the version 1.0b release, extensive use was made of the C preprocessor (CPP) to select included physics, geometry, problem size, dimensionality, and parallel execution at compilation time. In version 2, the only CPP macros which remain are those which expose MPI and MGMPI functions to the compiler and a macro ("PROBLEM") identifying the name of the problem-generator subroutine. Physics, geometry, and dimension attributes are now set by integer and logical

control parameters; Fortran 90 ALLOCATE statements are used to allocate field arrays at execution. As a result, **ZEUS-MP can now be reconfigured for problem size, dimensionality, included physics, and parallel topology at run time without recompilation.**

BUGS in Version 1.0b!!

ZEUS-MP v1.0b contains (at least) two undocumented and potentially severe bugs in the source code:

1. Subroutine tranx2.F: values of the cpp macro UNROLL_J other than 4 will produce incorrect source code.

2. Subroutines tranx2.F and tranx3.F: the "xi" expressions are incorrect for curvilinear coordinates. Correct expressions can be found in corresponding subroutines in Version 2.

Numerical Overview

The overall numerical scheme adopted by ZEUS-MP is the same as that employed by earlier ZEUS codes. An operator-split scheme is adopted whereby in the "source step", fluid velocities and internal energies are updated due to the effects of body forces, artificial viscosity, local heating/cooling/transport due to radiation diffusion (if applicable), and PdV work. In the subsequent "transport step," these fluid velocities are used to advect the field variables (density, internal energies, composition) through the computational mesh. A three-level listing of the subroutine calls made during a hydro cycle is as follows:

- call GRAVITY (evaluate gravitational potential)

  If 1D --> use GM/r
  If 2D --> use GM/r OR call GRAV2D_CG (conjugate gradient Poisson Solver)
  If 3D --> call GRAV3D_CG OR call MGMPI (conjugate gradient or Multigrid solvers)

- call SRCSTEP (execute "source" step)

  - call EOS_D (compute pressure)
  - call FORCES_D (update velocities due to body forces)
  - call AVISC_D (update velocities and gas energy due to artificial viscosity)
  - call RADUPDT (perform radiation diffusion, if desired)
  - call PDV_D (add PdV work term to internal energy)

- call TRANSPRT (execute "transport" -- by which we really mean "advection" -- step)

  - call HSMOC (update magnetic fields via the Hawley-Stone Method of Characteristics)

  - call ADVX1 (advect along "x1" axis)

    - call TRANX1 (advect density, gas/radiation energies, abundances)
    - call MOMX1 (advect momentum densities)

  - call ADVX2 (advect along "x2" axis)

        - call TRANX2 (advect density, gas/radiation energies, abundances)
        - call MOMX2 (advect momentum densities)

    - call ADVX3 (advect along "x3" axis)

        - call TRANX3 (advect density, gas/radiation energies, abundances)
        - call MOMX3 (advect momentum densities)

- call NUDT (compute timestep for next hydro cycle)

NOTE: The order in which the ADVX* subroutines are called is cyclically permuted from one timestep to the next for numerical stability.

Implementing Parallelism

ZEUS-MP uses the *Message-Passing Interface* (MPI) to achieve parallelism in a distributed-memory model. ZEUS-MP uses spatial *domain decomposition* to subdivide and distribute the computational domain across the processors in a multi-process calculation. The domain can be subdivided along any or all of the computational axes which are "active" (i.e. have more than one physical zone) in a calculation.

Because ZEUS-MP adopts a distributed-memory approach, message-passing via MPI must be used to update the information in the "ghost zones" bounding a processor's share of physical zones. In order to facilitate the overlap of communication and computation, *asynchronous message passing* is employed with **MPI_ISEND** and **MPI_IRECV** function calls. These function calls are interleaved with Fortran DO loops which do actual computational work according to the following strategy:

1. Initiate send/receive requests for boundary data to/from neighboring processors.
2. Execute work on "interior" zones in local domain.
3. Verify that MPI_ISEND/MPI_IRECV processes have finished.
4. Execute work on boundary zones in local domain.

In a 3-D calculation, ghost "faces" must be exchanged along 3 coordinates, and for corner ghost zones to be updated correctly, the updates along axis "n" must complete before the updates along axis "n+1" are initiated. Therefore, the strategy outlined above is applied along *each* active coordinate axis, which means that the "interior" domain is subdivided into several pieces so that the communication/computation overlap strategy may be used along each axis. Subroutines that employ this strategy include:

1. EOS_D
2. FORCES_D
3. AVISC_D
4. PDV_D
5. ADVX1
6. ADVX2
7. ADVX3
8. NUDT

In addition to enhancing parallel efficiency, this "interleaving" strategy makes the subroutines listed above quite complicated in structure. For example, examination of ADVX1 (source file advx1.F) shows multiple occurrences of the TRANX1 and MOMX1 subroutine calls, each operating over a different range of zone indices, interspersed with calls to BVAL* subroutines which do the heaving lifting of MPI. On scalable hardware architectures, however, this strategy has proven quite effective at promoting parallel efficiency.