User Guide for AMR Ideal MHD Code Code

Dan Martin and Phil Colella and Ravi Samtaney
Applied Numerical Algorithms Group

October 6, 2005

1 Overview

The AMR Ideal MHD code implements the algorithm described in [2] using the software design described in the "AMR Ideal MHD Software Design" [?].

2 Obtaining and compiling the code

The AMR Ideal MHD code may be obtained from the ANAG NASA CAN website (`http://davis.lbl.gov/NASA`); using the code also requires the Chombo framework [1]. The Chombo framework also uses HDF5 for I/O.

To compile the code, first install the HDF5 and Chombo libraries according to the instructions in each package. Change to `Chombo/example/PPMAMRGodunov/execIdealMHD` . Finally, compile the code: `make all [DIM=<2,3>] [DEBUG=<TRUE,FALSE>]` with "DEBUG=FALSE" producing optimized code.

An executable of the form `amrGodunov<DIM>d.<config-string>.ex` is produced, where the `<config-string>` contains information about how the code was compiled.

3 Running the code

To run the code, type `amrGodunov<DIM>d.<config-string>.ex <inputs-file>` , where `inputs-file` is a file containing the parameters needed to specify the run parameters.

3.1 Inputs file options

The format for the inputs file is generally of the form `<group>.<variable> = <value(s) >` , where `<group>` generally indicates what part of the code uses a given input. Everything following a "#" on a line is ignored, and in the case of multiple instances of a variable in an inputs file, the last instance is used. A sample inputs file is in `Chombo/example/PPMAMRGodunov/execIdealMHD/explosion.inputs`.

Some input parameters are required, while others have default values if they are not specified in the inputs file. Required variables are listed first, followed by optional ones.

3.1.1 Input parameters for `main`

- `godunov.problem` (required) string – name of problem type to solve. Current implemented problems are "rotor", "wave", or "explosion".
- `godunov.gamma` (required) Real – ratio of specific heats.
- `godunov.max_step` (required) integer – maximum number of timesteps to compute.
- `godunov.max_time` (required) Real – maximum solution time to compute to.
- `godunov.domain_length` (required) Real – physical size of the longest dimension of the domain.
- `godunov.num_cells` (required) SpaceDim integers – number of cells in each direction in the base computational domain.
- `godunov.is_periodic` (required) SpaceDim integers. In each coordinate direction, if 1, domain is periodic in that direction; if 0, non-periodic.
- `godunov.max_level` (required) integer – finest allowable refinement level. 0 means there will be no refinement.
- `godunov.ref_ratio` (required) `max_level` integers (one for each level). refinement ratios between levels. First number is ratio between levels 0 and 1, second is between levels 1 and 2, etc.

- `godunov.regrid_interval` (required) `max_level` integers (one for each level) – number of timesteps to compute between regridding. A Negative value means there will be no regridding.
- `godunov.tag_buffer_size` (required) integer – amount by which to grow tags (as a safety factor) before passing to MeshRefine.
- `godunov.refine_thresh` (required) Real – the threshold value of the undivided gradient of the density at which we tag cells for refinement during the grid generation process. if the undivided gradient is greater than the threshold, then the cell is tagged for refinement to a finer level.
- `godunov.block_factor` (required) integer – the block factor is the number of times that grids will be coarsenable by a factor of 2. This can have implications on how well multigrid solvers work. A higher number produces "blockier" grids.
- `godunov.max_grid_size` (required) integer – the largest allowable size of a grid in any direction. Any boxes larger than that will be split up to satisfy this constraint.
- `godunov.fill_ratio` (required) Real between 0 and 1. – the efficiency of the grid generation process. Lower number means that more extra cells which aren't tagged for refinement wind up being refined along with tagged cells. The tradeoff is that higher fill ratios lead to more complicated grids, and the extra coarse-fine interface work may outweigh the savings due to the reduced number of fine-level cells.
- `godunov.normal_predictor` (required) string (either "PPM" or "PLM") – if this is "PPM", use the piecewise parabolic method (PPM) for the hyperbolic normal predictor. If "PLM", use the Piecewise linear method (PLM).
- `godunov.use_fourth_order_slopes` (required) integer. If 1, use 4th-order slopes for the normal predictor, Otherwise, use second-order slopes.
- `godunov.use_prim_limiting` (required) integer. If 1, limit slopes in primitive variables.

- `godunov.use_char_limiting` (required) integer. If 1, do slope limiting using characteristic variables. Only one of `prim_limiting` or `char_limiting` may be turned on, or no limiting at all may be used.
- `godunov.use_flattening` (required) integer – If 1, do slope flattening.
- `godunov.use_artificial_viscosity` (required) integer – if 1, use artificial viscosity.
- `godunov.artificial_viscosity` (required) Real – artificial viscosity.
- `godunov.filter_BField` (required) integer – if 1, filter the magnetic field.
- `godunov.cfl` (required) Real – CFL number (maximum allowable value for max(vel)*dt/dx.
- `godunov.initial_cfl` (required) Real – safety factor to multiply the initial timestep by.
- `godunov.max_dt_growth` (required) Real – maximum factor by which the timestep can increase from one timestep to the next.
- `godunov.dt_tolerance_factor` (required) Real – Let the time step grow by this factor above the "maximum" before reducing it
- `godunov.verbosity` (default = 0) integer – higher number results in more verbose text output.
- `godunov.checkpoint_interval` (default = 0) integer – number of timesteps between writing checkpoint files. Negative number means that checkpoint files are never written, 0 means that checkpoint files are written before the initial timestep and after the final one.
- `godunov.plot_interval` (default = 0) integer – number of timesteps between writing plotfiles. Negative number means that plotfiles are never written, 0 means that plotfiles are written before the initial timestep and after the final one.
- `godunov.fixed_dt` (default = -1) Real – if positive, code will use this value for the timestep.

3.1.2 Explosion Problem-specific inputs

Different values for `problem` in the inputs file will require different specific input specifications to define the problem. The explosion problem has these additional inputs:

- `godunov.density_jump` (required) Real – $\frac{\rho}{\rho_0}$ inside the explosion.
- `godunov.pressure_jump` (required) Real – $\frac{p}{p_0}$ inside the explosion.
- `godunov.p0` (required) Real – ambient pressure outside explosion.
- `godunov.rho0` (required) Real – ambient density outside explosion.
- `godunov.B_direction` (required) integer – coordinate direction of initial magnetic field.
- `godunov.B_magnitude` (required) Real – magnitude of (spatially-constant) initial magnetic field.
- `godunov.initial_center` (required) Real – location of center of explosion.
- `godunov.initial_size` (required) Real – initial radius of explosion.
- `godunov.initial_velocity` (required) Real – initial background (spatially constant) velocity field.

3.1.3 Rotor Problem-specific inputs

The rotor problem has no additional inputs.

3.1.4 Wave Problem-specific inputs

The plane wave ("wave") problem has these additional inputs:

- `godunov.alpha` (required) Real – Angle between wave and horizontal.
- `godunov.pdir` (required) integer – Eigenvalue perturbation direction.
- `godunov.kratio` (required) integer – Ratio between x- and y- wavenumbers.

- `godunov.waveNumber` (required) integer – wavenumber in the $x-$direction.
- `godunov.pertAmplitude` (required) Real – perturbation amplitude.

3.2 Visualizing the results

If `godunov.plot_interval` is non-negative, the AMR Ideal MHD code will write solutions out into HDF5 plotfiles, which are in the format used by the `ChomboVis` visualization tool.

References

[1] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. unpublished, 2000.

[2] R Samtaney, P Colella, T J Ligocki, D F Martin, and S C Jardin. An adaptive mesh semi-implicit conservative unsplit method for resistive MHD. In *SciDAC 2005 proceedings*, volume 16 of *Journal of Physics::Conference Series*. Institute of Physics Publishing, 2005.