User Guide for AMR Self-Gravity Code Code

Dan Martin and Phil Colella and Francesco Miniati
Applied Numerical Algorithms Group

October 17, 2005

1 Overview

The AMR Self-Gravity code implements the algorithm described in [2, 3] using the `AMRTimeDependent` infrastructure in the `Chombo` library package.

2 Obtaining and compiling the code

The AMR Self-Gravity 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/AMRSelfGravity/exec` . 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/AMRSelfGravity/exec/selfGravity.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`

- `charm.problem` (required) string – name of problem type to solve. Current implemented problems are "dustcollapse", "gastests", or "ramp".
- `charm.max_step` (required) integer – maximum number of timesteps to compute.
- `charm.max_time` (required) Real – maximum solution time to compute to.
- `charm.num_cells` (required) SpaceDim integers – number of cells in each direction in the base computational domain.
- `charm.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.
- `charm.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.
- `charm.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).
- `charm.use_prim_limiting` (required) integer. If 1, limit slopes in primitive variables.

- `charm.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.

- `charm.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 efficiently multigrid solvers work. A higher number produces "blockier" grids.

- `charm.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.

- `charm.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.

- `charm.max_dt_growth` (required) Real – maximum factor by which the timestep can increase from one timestep to the next.

- `charm.dt_tolerance_factor` (required) Real – Let the time step grow by this factor above the "maximum" before reducing it

- `charm.bc_lo` (required) SpaceDim integers – integers specifying the type of boundary conditions to use on the low side in each direction: 0 is Dirichlet, 1 is Neumann, 2 is infinite-domain boundary conditions, and 3 is Gauss BC's. Note that Gauss BC's are based on Gauss's theorem are implemented specifically for sperically symmetric mass distributions.

- `charm.bc_hi` (required) SpaceDim integers – integers specifying the type of boundary conditions to use on the low side in each direction: 0 is Dirichlet, 1 is Neumann, 2 is infinite-domain boundary conditions, and 3 is Gauss BC's. Note that Gauss BC's are based on Gauss's theorem are implemented specifically for sperically symmetric mass distributions.

- `charm.verbosity` (default = 0) integer – higher number results in more verbose text output.
- `charm.gamma` (default = 1.667) Real – ratio of specific heats.
- `charm.Rs_tolerance` (default = 1.0e-6) Real – tolerance for error in Riemann solver.
- `charm.max_rs_iter` (default = 10) integer – maximum number of iterations in the Riemann solver.
- `charm.max_mach` (default = 50) Real – Threshold for switch from energy to entropy conservation equation.
- `charm.artificial_viscosity` (default = 0.0) Real – Artificial viscosity (0 means no artificial viscosity is used).
- `charm.domain_length` (default = 1.0) Real – physical size of the longest dimension of the domain.
- `charm.is_periodic` (default = 0's) SpaceDim integers. In each coordinate direction, if 1, domain is periodic in that direction; if 0, non-periodic.
- `charm.max_level` (default = 0) integer – finest allowable refinement level. 0 means there will be no refinement.
- `charm.tag_buffer_size` (default = 3) integer – amount by which to grow tags (as a safety factor) before passing to MeshRefine.
- `charm.max_init_ref_level` (default = 0) integer – maximum level at initial startup.
- `charm.use_gradient_refine` (default = 0) integer – tag on undivided relative gradients ($\frac{\Delta x}{Avg(\phi)}\frac{\partial \phi}{\partial x}$) for regridding.
- `charm.grad_refine_thresh` (default = 100) Real – threshold number for tagging cells for refinement based on gradients.
- `charm.grad_var_interv` (default (0,0) 2 integers – interval of components in state vector to use for tagging based on gradient.

- `charm.use_shock_refine` (default = 0) integer – tag on shocks (pressure jumps) for regridding.
- `charm.pres_jump_thresh` (default = 0) Real – threshold number for tagging cells for refinement based on pressure jumps.
- `charm.use_over_dense_refine` (default = 0) integer – tag on overdense regions.
- `charm.cell_mass_thresh` (default = 100) Real – threshold for use with overdense tagging.
- `charm.use_jeans_refine` (default = 0) integer – tag on Jeans length
- `charm.jeans_resol_thresh` (default = 4) Real – lowest resolution threshold for use with Jeans length tagging.
- `charm.use_fourth_order_slopes` (default = 1) integer. If 1, use 4th-order slopes for the normal predictor, Otherwise, use second-order slopes.
- `charm.use_flattening` (default = 1) integer – If 1, do slope flattening.
- `charm.use_artificial_viscosity` (default = 1) integer – if 1, use artificial viscosity.
- `charm.artificial_viscosity` (default = 0.1) Real – artificial viscosity.
- `charm.cfl` (default = 0.8) Real – CFL number (maximum allowable value for max(vel)*dt/dx.
- `charm.initial_cfl` (default = 0.1) Real – safety factor to multiply the initial timestep by.
- `charm.force_stencil` (default = 0) integer – Typeof stencil to use: 0 = 2 points, 1 = 4 points, 2 = 10 points. Note that the four-point stencil is not supported by the coarse-fine interpolation operators, so the 4-point stencil may not be used for AMR computations.
- `charm.use_delta_phi_corr` (default = 0) integer – use correction for $\Delta\phi$.

- `charm.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.
- `charm.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.
- `charm.plot_prefix` (default = "pltstate") string –
- `charm.chk_prefix` (default = "chk") string –
- `charm.fixed_dt` (default = -1) Real – if positive, code will use this value for the timestep.
- `charm.fixed_hierarchy` string – If this is present in the inputs file, code will run with a fixed AMR hierarchy specified in the string argument.
- `charm.restart_file` string – If this is present in the inputs file, the code will restart from the file specified in the string argument.

3.1.2 Dust-collapse Problem-specific inputs

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

- `charm.cloud_radius` (default = 0.5) Real – Initial radius of dust cloud.
- `charm.cloud_density` (default = 1.0) Real – Initial density of dust cloud.

3.1.3 gastests Problem-specific inputs

The "gastests" problem has these additional inputs:

- `charm.gas_test` (required) string – Type of test to run; can be "explosion" or "wave".

- `charm.shock_mach` (default = 10.0) Real – Mach number of explosion test problem.
- `charm.size` (default = 0.25) Real – size of explosion.
- `charm.center` (default = zeros) SpaceDim Reals – center of initial condition for explosion or wave problem.
- `charm.velocity` (default = zero) SpaceDim Reals – initial velocity of the gas.

3.1.4 ramp Problem-specific inputs

The "ramp" problem has these additional inputs:

- `charm.angle_deg` (default = 30.0) Real – angle of ramp.
- `charm.shock_mach` (default = 10.0) Real – initial magnitude of shock.
- `charm.xcorner` (default = 0.1) Real – location of left-corner of ramp.

3.2 Visualizing the results

If `charm.plot_interval` is non-negative, the AMR Self-Gravity 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] Dan Martin and Phil Colella. Self-gravity amr algorithm specification. available at http://davis.lbl.gov/NASA, 2004.

[3] F. Miniati and P Colella. Block structured adaptive mesh and time refinement for hybrid, hyperbolic, and N-body systems. in preparation.