FLASH UHD Documentation

Christopher C. Lindner, Miloš Milosavljević,

Department of Astronomy, University of Texas, 1 University Station C1400, Austin, TX 78712

ABSTRACT

The purpose of this documentation is to document the FLASH unsplit hydrodynamics solver and the changes needed to add radiation hydrodynamics to this solver.

1. Introduction

The latest version of the FLASH code is available in my directory
`/data1/r900-1/lindner/FLASH4.2.1/source/`

Most of the files relevant to the unsplit solver are in
`/data1/r900-1/lindner/FLASH4.2.1/source/physics/Hydro/HydroMain/unsplit/Hydro_Unspl`

Any file or directory names I mention will be relative to this directory.

Section 2 contains a basic overview flowchart of how the solver works. There are some brief descriptions of select files as well.

2. General Flowchart

Hydro Main FLASH hydro driver that calls the unsplit solver.

hy_uhd_unsplit This basically runs through the entire unsplit solver. It calls the following functions (`hy_uhd_` prefixes have been ommitted):

putGravityUnsplit

getRiemannState Calculates and stores Riemann state values at cell facethem so we can use these to compute fluxes, see MC 4.2.3

- First, some "‘hybrid order"’ things that we do not use (?).
- Then, slope flattening is carried out: MC 4.2.2 .
- Then, we start calculating Riemann states.
- **dataReconstOneStep** Evolves cell-centered values by $\Delta t/2$ at cell interfaces using PPM characteristic tracing. Most of this is carried out in the function below.

- **DataReconstructNomralDir_PPM** The driver of PPM and characteristic tracing. The steps described are as follows:
  1. Calculate $\vec{\lambda}$, which is the vector of characteristic speeds.
  2. Projects $\bar{\Delta}$ to primitives (?). Carries out $U = \frac{1}{2}\left(U_C + U_L - \bar{\Delta}_O - \bar{\Delta}_N\right)$. I'm not really sure what this all means.
  3. Carries out steepening (which is turned off in the setup we use).
  4. Flattening.
  5. Monotonicity Check (?).
  6. Tries to guess at the interface states (polynomial fitting??).
  7. Performs characteristic Tracing. I have more detailed notes on what happens during this, but it's similar to MC 4.2.3 . See Section 3.
  8. Finalizes the calculation of Riemann states?

  These functions are called during this:
  - **eigenParameters** Calculates several parameters needed for the wave speed eigenvalue calculations.
  - **eigenValue** Calculates the eigenvalues (wave speeds) that fill the eigenvectors.
  - **eigenVector** Fills the $\vec{\lambda}$ eigenvector in primative or conservative form.
  - **upwindTransverseFlux**
- Applies geometric terms.
- Updates Gravity.
- Stores scratch terms for each direction.
- Applies transverse correction terms for 3D.
- **upwindTransverseFlux**

getFaceFlux Computes fluxes at cell faces. Also calls specific solvers (e.g. **HLL**). Calculates min timestep and alters fluxes for artificial viscosity.

- **HLL** This is the most stable solver we've been using. It's in charge of computing the high-order Godunov fluxes based on the L and R Riemann states.
  1. Converts primitives to conserved variables for left and right states.
     prim2con Calculates conversions from primitive to conserved variables. The conserved variables are $\rho, \rho\vec{v}, E_{\rm tot}$
     prim2flx Converts from variables to fluxes. $F = \rho v$, $F = \rho v v + P$, and $F = v(E+P)$.
  2. Calculates $F^* = [S_R F_L - S_L F_R + S_R S_L (U_R - U_L)] / (S_R - S_L)$

unsplitUpdate Given the fluxes, updates the conserved, cell-centered quantities. This is responsible for getting all the variables and geometeric terms in the right form so it can call two subfunctions that are present in this file.

- **updateConservedVariable** Computes $U = U - \frac{\Delta t}{\Delta x}(F_R - F_L)$.
- **updateInternalEnergy** When flag `hy_useAuxEintEqn` is set, computes $\epsilon = \epsilon + \frac{\Delta t}{\Delta x}(F_L - F_R + P(F_L - F_R))$.

multiTemp/unsplitUpdateMultiTemp Updates energyies for the special 3T conditions. See the FLASH manual for more information on the "‘Rage-like"’ approach used here. Some of the weird uhd crashes I’ve seen occur here. There are comments that explicitly state these crashes occur for "‘unknown reasons."’ This calls **hy_uhd_ragelike**.

energyFix Corrects energy in cases where the total energy is advected and `eint` can become negative. In the three temperature case, this step is actually handled by **unsplitUpdateMultiTemp**.

Grid_conserveFluxes

Eos_wrapped

putGravityUnsplit

addGravityUnsplit

energyFix

multiTempAfter This is only needed for when you are using MHD and three temperature, I think.

3. Characteristic Tracing

These are equations I was able to decifer from `hy_uhdDataREconstructNormalDir_PPM`. I’m going to focus on step 8, characteristic tracing, which is similar to MC section 4.2.3 .

$$\lambda_0 = \begin{pmatrix} v_x - c \\ v_x \\ v_x \\ v_x + c \end{pmatrix} \tag{1}$$

Where c is just the sound speed in hydro. This is the eigenvector computed in the `hy_uhd_eigenVector` function.

$$\mathbf{Q} = \begin{pmatrix} \rho \\ v_x \\ v_y \\ v_z \\ P \end{pmatrix} \tag{2}$$