README (draft)

Brandon Gusto

February 23, 2021

Overview

The hybrid adaptive multiresolution (HAMR) code is an add-on package for the FLASH simulation code (Fryxell et al. 2000), and can be downloaded freely at https://github.com/blg13/HAMR-FLASH. The HAMR approach uses multiresolution (MR) indicators not only to guide the refinement/coarsening of Cartesian meshes utilizing adaptive mesh refinement (AMR), but also to accelerate calculations of physical quantities in smooth regions. Note that the current implementation is limited to the Paramesh library (MacNeice et al. 2000), and that the code currently assumes that cells that are equally spaced in each direction. The package is designed to patch the latest release of FLASH, `FLASH4.6.2`.

Description

The HAMR-FLASH package is divided into two components: multiresolution-driven AMR (MrAMR) and HAMR. The first component can be used if one wants to use MR indicators for AMR purposes only. The latter component uses the former one, but also introduces solver-adaptivity; that is, using MR indicators to further accelerate the calculation of physical quantities using interpolation from coarser levels. The physical quantities currently considered by the algorithm are the hydrodynamic fluxes (split PPM), equation of state (EoS), and the reactive source terms.

The AMR blocks in FLASH are tagged for refinement or coarsening according to MR smoothness indicators (hereafter referred to as detail coefficients). A new directory, `source/Grid/GridSolvers/Multiresolution`, is included to handle the calculation of detail coefficients, and the construction of the MR mask. The MR mask is constructed by ignoring the cells whose corresponding detail coefficients are smaller than a user-defined tolerance. In general the construction of the mask requires communication between blocks, as the algorithm 'warns' cells surrounding non-smooth features. The calling of MR subroutines is handled by modified Paramesh files, located in

source/Grid/GridMain/paramesh/multiresolution. Runtime parameters associated with this module are listed below:

- wvlt_thresh (real, 0.e0): user-defined tolerance for truncating detail coefficients
- wvlt_userelative (boolean, false): computes detail coefficient values relative to solution magnitudes
- wvlt_nbuffer (integer, 1): number of cells in each direction of the MR buffer region
- wvlt_maxlvl (integer, 3): number of levels in the local MR hierarchy (minimum of three required for AMR)
- wvlt_diagnostic (boolean, true): saves detail coefficient values for checkpoint and plotfiles
- wvlt_refine_var_1 (string, "none"): refinement variable
- wvlt_refine_var_2 (string, "none"): refinement variable
- wvlt_refine_var_3 (string, "none"): refinement variable
- wvlt_refine_var_4 (string, "none"): refinement variable
- wvlt_refine_var_5 (string, "none"): refinement variable
- wvlt_refine_var_6 (string, "none"): refinement variable

The interpolation of numerical flux follows the procedure originally introduced by Harten (Harten 1994). High quality numerical flux calculations are replaced in smooth regions (as determined by the tolerance) with interpolation from interfaces corresponding to coarser grid levels, where flux values are already available. The same general approach is used for the evaluation of the Helmholtz EoS, and integration of reaction networks for reactive problems. Runtime parameters associated with these modules are listed below:

- wvlt_threshfactor (real, 0.e0): safety factor pre-multiplying the parameter wvlt_thresh for the construction of the solver-adaptive mask
- wvlt_interphydro (boolean, false): allows for the interpolation of numerical fluxes
- wvlt_interpeos (boolean, false): allows for the interpolation of EoS
- wvlt_interpburn (boolean, false): allows for the interpolation of reaction products

Note that shortcuts are included for convenience when setting up and compiling new problems:

- amr_wvlt: enables MR-driven AMR
- mrppm: includes source/physics/Hydro/HydroMain/split/PPM/mrPPMKernel and enables interpolation of numerical fluxes

Dependencies

The software required to use the HAMR-FLASH package is as follows:

1. A copy of `FLASH4.6.2` (earlier releases are not yet supported)
2. `perl` (`v5.16.0` or later recommended)

Installation

How to install the HAMR package:

1. Download the package at https://github.com/blg13/HAMR-FLASH
2. Navigate to the root directory of your copy of `FLASH4.6.2`
3. Run the included installer script, `install_patch.pl`

Once these steps are successfully completed, the modified FLASH code is ready to run.

Examples

Blast2

An example setup utilizing MR-driven AMR and solver-adaptivity for the Blast2 problem is given below:

```
$PATHTOFLASH/setup Blast2 \
   -auto \
   -1d \
   -nxb=16 \
   -maxblocks=2000 \
   +amr_wvlt \
   +mrppm \
   -objdir=object
```

A `flash.par` file is provided in `examples/blast2`, using a uniform resolution of `2048` cells and a solver-adaptive tolerance of `1e.-04`.

References

Fryxell, B., K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. 2000. “FLASH: An Adaptive

Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes" 131 (1) (November): 273–334. doi:10.1086/317361.

Harten, A. 1994. "Adaptive Multiresolution Schemes for Shock Computations." *Journal of Computational Physics* 115 (2): 319–338.

MacNeice, Peter, Kevin M. Olson, Clark Mobarry, Rosalinda de Fainchtein, and Charles Packer. 2000. "PARAMESH: A Parallel Adaptive Mesh Refinement Community Toolkit." *Computer Physics Communications* 126 (3): 330–354. doi:https://doi.org/10.1016/S0010-4655(99)00501-9. https://www.sciencedirect.com/science/article/pii/S0010465599005019.