Making Your Own Application I: The Simulation Directory

Dr. Lynn B. Reid

June 23, 2009

An Advanced Simulation & Computing (ASC)
Academic Strategic Alliances Program (ASAP) Center
at The University of Chicago

Outline

- The Simulation directory
  - Standard contents
  - Alternative routines
- Customizing FLASH for a new simulation
  - Config file creation
- Developing for fame and glory

The Simulation Unit

- Typical Unit, obeys architecture, naming conventions, inheritance, etc. rules.
- Special Unit in that it always “wins” inheritance and parameter wars.
- FLASH problems is defined by directories in FLASH3 /source/Simulation/SimulationMain.
- The Simulation directory gives people working on a particular problem a place to put problem specific code that replaces the default functionality in the main body of the code
- It’s also a place to tell the setup script which units this problem will need from the rest of the code

What's in the Simulation Directory?

- Normal UnitMain implementation requirements
  - Simulation_data, Simulation_init, (Simulation_finalize), Simulation_initBlock
  - Makefile (with usually Simulation_data only)
  - Config file
  - Possibly other API functions: e.g. Simulation_initSpecies
- Specific to simulations:
  - Parameter files flash.par, testUG.par, etc.
  - Replacements for routines located elsewhere in directory tree
  - Routines that implement local functions e.g. sim_derivedVariables.F90

Required Code for a New Simulation

- There are certain pieces of code that all simulations must implement:
  - Simulation_data.F90: Fortran module which stores data and parameters specific to the Simulation.
  - Simulation_init.F90: Reads the runtime parameters, and performs other necessary unit initializations.
  - Simulation_initBlock.F90: Sets initial conditions in a single block.
- Optionally, a simulation could implement:
  - Simulation_initSpecies.F90: To give the properties of the species involved in a multispecies simulation

Customized Code for a new Simulation

- In a FLASH simulation directory, you can place code that overrides the functionality you would pick up from other code units
- In the custom code you can modify:
  - Boundary conditions (Grid_applyBCEdge.F90)
  - Refinement criterion (Grid_markRefineDerefine.F90)
  - Diagnostic integrated quanties for output (in the flash.dat file), e.g., total mass (a default) or vorticity (IO_writeIntegralQuantities.F90)
  - Diagnostics to compute new grid scope variables (Grid_computeUserVars.F90)
- In general, this is a place to hack the code in ways specific to your problem, and you can hack basically anything

SimulationComposition SubUnit

- Default code for setting up properties of multiple species
  - Simulation_initSpecies.F90
  - Usually calls routines like Multispecies_setProperty
- Documented in Multispecies chapter and Robodoc headers
- Has two implementations: Burn and Ionize
  - Burn uses text file SpeciesList.txt to initialize isotopes
  - Ionize does more fractionation
  - If developing your own, follow Burn model for simplicity

Example Non-trivial Setup: TwoGamma

- TwoGamma is for a simple test of advecting two fluids having different gammas to investigate whether an instability develops at the interface between the two fluids.
- This Simulation implements:
  - Simulation_initBlock.F90, per usual
  - Simulation_initSpecies.F90, because it has multiple fluids
  - and Grid_applyBCEdge.F90, because it needs custom boundary conditions on the lower x edge of the domain

TwoGamma Config File

```
# configuration file for the TwoGamma target problem

REQUIRES Driver
REQUIRES physics/Hydro
REQUIRES physics/Eos/EosMain/Multigamma
REQUIRES Multispecies
REQUESTS IO

# Parameters

D sim_p0   constant pressure
PARAMETER sim_p0        REAL      2.5e-0

D sim_rho1  density of the first fluid
PARAMETER sim_rho1      REAL      1.0e-0

D sim_rho2  density of the second fluid
PARAMETER sim_rho2      REAL      1.0e-0

D sim_cvelx  initial velocity
PARAMETER sim_cvelx     REAL      0.1e-0

SPECIES FLD1
SPECIES FLD2
```

TwoGamma flash.par

```
# AMR parameters
lrefine_max = 4
lrefine_min = 4

# simulation parameters
basenm   = "twogamma_"
restart  = .false.
plotFileIntervalTime       = 0.1
checkpointFileIntervalTime = 0.5
nend     = 15000
tmax     = 10.0
checkpointFileNumber = 0
plotFileNumber = 0

dtini = 1.e-10
dtmin = 1.e-10

cfl = .5
cvisc = .1
```

TwoGamma flash.par (Cont.)

```
smlrho = 1.e-10
smallt = 1.e-10

xmin = 0.0e0
xmax = 1.0
ymin = 0.0e0
ymax = 1.0e0

geometry = "cartesian"

# variables for plotting
plot_var_1 = "dens"
plot_var_2 = "temp"
plot_var_3 = "ener"
plot_var_4 = "pres"
plot_var_5 = "velx"
plot_var_6 = "fld1"
plot_var_7 = "fld2"

xl_boundary_type = "user"
xr_boundary_type = "outflow"
yl_boundary_type = "periodic"
yr_boundary_type = "periodic"
```

TwoGamma Simulation_initSpecies.F90

```
subroutine Simulation_initSpecies()

  implicit none
#include "Multispecies.h"
#include "Flash.h"
#include "Multispecies.h"
#include "Multispecies_interface.h"


  call Multispecies_setProperty(FLD1_SPEC, A, 1.)
  call Multispecies_setProperty(FLD1_SPEC, Z, 1.)
  call Multispecies_setProperty(FLD1_SPEC, GAMMA, 1.66666666667e0)

  call Multispecies_setProperty(FLD2_SPEC, A, 4.0)
  call Multispecies_setProperty(FLD2_SPEC, Z, 2.0)
  call Multispecies_setProperty(FLD2_SPEC, GAMMA, 2.0)

end subroutine Simulation_initSpecies
```

TwoGamma Grid_applyBCEdge.F90

```
if(face==LOW) then
   select case (bcType)
   case(OUTFLOW)
      do i = 1,guard
         dataRow(i)= dataRow(guard+1)
      end do
   case(USER_DEFINED)
      select case(var)
      case(GAMC_VAR)
         dataRow(1:guard)=sim_gammac1
      case(DENS_VAR)
         dataRow(1:guard)=sim_rho1
      case(PRES_VAR)
         dataRow(1:guard)=sim_p0
      case(VELX_VAR)
         dataRow(1:guard)=sim_cvelx
      case(VELY_VAR)
         dataRow(1:guard)=0.0
      case(VELZ_VAR)
         dataRow(1:guard)=0.0
      case(ENER_VAR)
         dataRow(1:guard)=max(0.5*(sim_cvelx**2)+sim_int1,sim_small)
```

Developing for Fame and Glory

- FLASH depends upon many contributions
  - Amalgamation of physics and computational research
  - 82 person-years of effort in current release!
- What does a contributor receive?
  - Your research reaches a wide audience
  - Contacts with FLASH community – jobs jobs jobs
  - Citations, citations, citations!
- What do you need to do?
  - Observe basics of style
  - Document! Document! Document!
  - Wait until after publication before release

Questions?