FLASH Code Tutorial

a user’s perspective

Robi Banerjee
Hamburger Sternwarte
banerjee@hs.uni-hamburg.de
Some background information:

- **3D AMR MHD multi-physics code**
- main code development at
  
  FLASH Center at the University of Chicago
  DOE NNSA-ASC OASCR Flash Center
  web: [http://flash.uchicago.edu](http://flash.uchicago.edu)
  
  currently about 20 people at the Flash Center
- contribution from many individual groups, e.g.
  - hybrid-characteristic ray-trace (Rijkhorst et al., Peters et al.)
  - sink particles (Federrath et al.)
  - Barns & Hut tree algorithm (Wunsch)
  - chemistry
Some background information:

- Research applications
  - thermonuclear flashes
  - high energy density physics (HEDP)
  - fluid-structure interaction

- star formation
- star-star & star-planets interactions (CE phase)
- cosmology
- galaxy & galaxy cluster simulations
- magnetic field amplification
- turbulence
- ...

FLASH Code
FLASH Code
FLASH Code

Some background information:

• current release 4.0
  ➔ user manual ~ 500 pages!

• a bit of history:
  • 1997: founding of the ASCI (Accelerated Strategy Computing Initiative*) FLASH Center
  ➔ main goal: thermonuclear flashes at neutron star and white dwarf surfaces

*now ASC: Advanced Simulation and Computing
• a bit of history:

• **2000**: first FLASH Code release
  
  
  ⟹ HD based on PROMETHEUS Code (PPM)

  (Fryxell, Müller & Arnett, 1989, MPI for Astrophysics)

  ⟹ no MHD

  ⟹ no self-gravity

  ⟹ based on PARAMESH AMR library (block structured)

  Berger 1982
  Berger & Oliger 1984
  Berger & Colella 1989

  ⟹ supported grid geometry:

  cartesian, cylindrical, spherical
FLASH Code

• a bit of history:

• 2002: major revised version 2.3
  ⟷ MHD solver (Powell, Roe, Linde, Gombosi, De Zeeuw, 1999)
    directional split, finite volume scheme
    +non-ideal terms
    reduction of $\nabla B$ errors by
      - truncation error method (Powell et al. 1999)
        similar to Dedner et al. 2002 hyperbolic cleaning
      - elliptic projection method (Brackbill & Barnes 1980)
  ⟷ self-gravity / Poisson solver (P. Ricker)
  ⟷ tracer particle
  ⟷ “dark matter” particle
  ⟷ cosmology
  ⟷ HDF5 output format
FLASH Code

• a bit of history:

  • **2005**: improved, long-lived version 2.5
  new external developments:
    ➞ ray-trace based radiation-transfer for point sources
      
      \[\text{(Rijkhorst, Plewa, Dubey, Mellema, A&A 2006, Peters, RB, Klessen, Mac Low 2010)}\]
      hybrid-characteristics

    ➞ **sink particles** \(\text{(Federrath, RB, Clark, Klessen, ApJ 2010)}\)
      - on the fly creation
      - mass accretion
      - interface for sub-grid feedback models (e.g. jets, SN, ...)

    ➞ **new MHD solvers** \(\text{(Waagan, Federrath, Klingenberg, JCP 2011)}\)
      - ensures positivity of density/pressure
FLASH Code

• a bit of history:

• 2008: completely restructured version 3
  ⟹ unsplit HD & MHD solver (*Donkwook Lee*).
  ⟹ support of more flexible grid structures
    e.g. uniform grid, PARAMESH 3(staggered grid),
  ⟹ different “database” calls
  ⟹ different directory structure
  ⟹ different structure of HDF5 output

  ⟹ constraint transport (CT) MHD on a staggered mesh
  ⟹ many new physical modules:
    - diffusion solver
    - energy deposition via LASER beam
FLASH code: Overview

• extensive user manual: flash4_ug.pdf

with hyperlinks to Flash center web site

3.4 Creating a Simulation init.F90

The routine Simulation init is called by the routine Driver initFlash at the beginning of the simulation. Driver initFlash calls Unit.init.F90 routines of every unit to initialize them. In this particular case, the Simulation init routine will get the necessary runtime parameters and store them in the Simulation.data Fortran module, and also initialize other variables in the module. More generally, all one-time initialization required by the simulation are implemented in the Simulation init routine.

FLASH Transition

In FLASH2, the contents of the if (.firstcall.) clause are now in the Simulation init routine in Flash4.

The basic structure of the routine Simulation init should consist of
FLASH code: Overview

• many example setups

25.1 Hydrodynamics Test Problems
  25.1.1 Sod Shock-Tube
  25.1.2 Variants of the Sod Problem in Curvilinear Geometries
  25.1.3 Interacting Blast-Wave Blast2
  25.1.4 Sedov Explosion
  25.1.5 Isentropic Vortex
  25.1.6 Wind Tunnel With a Step
  25.1.7 The Shu-Osher problem
  25.1.8 Driven Turbulence StirTurb
  25.1.9 Relativistic Sod Shock-Tube
  25.1.10 Relativistic Two-dimensional Riemann
FLASH code: Overview

• many example setups

25.2 Magnetohydrodynamics Test Problems
   25.2.1 Brio-Wu MHD Shock Tube
   25.2.2 Orszag-Tang MHD Vortex
   25.2.3 Magnetized Accretion Torus
   25.2.4 Magnetized Noh Z-pinch
   25.2.5 MHD Rotor
   25.2.6 MHD Current Sheet
   25.2.7 Field Loop
   25.2.8 3D MHD Blast

....

25.7 Other Test Problems
   25.7.1 The non-equilibrium ionization test problem
   25.7.2 The Delta-Function Heat Conduction Problem
   25.7.3 The HydroStatic Test Problem
   25.7.4 Hybrid-PIC Test Problems
   25.7.5 Full-physics Laser Driven Simulation
• most of the code: FORTRAN 90
• highly modular
  \[\Rightarrow\] fairly well organized source directory structure

• pre-compilation via \texttt{setup} python script
  \[\Rightarrow\] typical number of Fortran files : 900 (out of 3400)
  \[\Rightarrow\] typical number of code lines : 150,000
• hardware / computer specific makefiles in sites directory:

**Aliases**
- Prototypes/
- SEAS10927.gwu.edu/
- alc.llnl.gov/
- animal5/
- archimedes.uchicago.edu/
- bassi.nersc.gov/
- bgl.llnl.gov/
- bgl.mcs.anl.gov/
- bgl.sdsc.edu/
- bonsai.cfa.harvard.edu/
- brassica.asci.uchicago.edu/

...
FLASH code: Overview

• requirements
  • fortran compiler
  • MPI
  • HDF5
  • PARAMESH AMR library (delivered with FLASH source)
  • HYPRE (for implicit solvers, e.g. diffusion)

• output format
  • HDF5
  • NetCDF
  • some ASCII files
    log files, dat-files with integrated variables, ...
The grid structure:

• possible grid structures

  • **PARAMESH AMR** *(MacNeice et al. CPC 2000; Olson 2006)*
    - block structured AMR
    - current version 4.

  • **CHOMBO**
    patch based AMR, still under development

• uniform grid
  no AMR overhead
Block structured PARAMESH AMR:

- neighbouring blocks: max relative size: 2
- typically blocks of $8^D$ active grid cells
- effective resolution: $2^{l_{\text{max}}+2}$

($l_{\text{min}} = 1$)
Flash Code

Block structure:

- Typically blocks of $8^D$ active grid cells

- $\text{NBX, NBY, NBZ} \implies$ setup parameter

- + 2 to 4 ghost / guard cells:

  \[
  \text{n\text{guard}}
  \]

  dependent on order of the integration scheme:

  - PPM: $\text{n\text{guard}} = 4$
  - 8W MHD: $\text{n\text{guard}} = 2$
Block structured PARAMESH AMR:

• setup up of non-cubic simulation box with cubic blocks:

  \[ \text{\Rightarrow nblockx, nblocky, nblockz} \]

  \[ = \text{number of level-1-blocks in x,y,z direction} \]

Example: \( \text{nblockx} = 4, \text{nblocky} = 1 \)

  \( \text{xmin} = 0, \text{xmax} = 4, \text{ymin} = -0.5, \text{ymax} = 0.5 \)
AMR: refinement / de-refinement

• standard criterion: second derivative criterion
  ⇒ “shock” capturing criterion
  ⇒ test with dimensionless error: \(|d^2f/dx^2| / |f|/(dx)^2\)

• refine on user defined variables (4 possible)
  refine_var_1 = “dens”

  set in flash.par

• parameter:
  refine_cutoff_1 = 0.8
  derefine_cutoff_1 = 0.2
AMR: refinement / de-refinement

• further criterions
  • Jeans criterion
  • refine on particle position

in

Particles/ParticlesMain/active/Sink/
Particles_sinkMarkRefineDerefine.F90

• implementing own criterion:

copy gr_markRefineDerefine.F90
in own setup directory

⇒ overrides standard routine
<table>
<thead>
<tr>
<th><strong>boundary_type</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>periodic</td>
<td>Periodic (‘wrap-around’)</td>
</tr>
<tr>
<td>reflect</td>
<td>Non-penetrating boundaries; plane symmetry, the normal vector components change sign</td>
</tr>
<tr>
<td>outflow</td>
<td>Zero-gradient boundary conditions; allows shocks to leave the domain</td>
</tr>
<tr>
<td>diode</td>
<td>like outflow, but fluid velocities are never allowed to let matter flow into the domain: normal velocity components are forced to zero in guard cells if necessary</td>
</tr>
<tr>
<td>axisymmetric</td>
<td>like reflect, but both normal and toroidal vector components change sign. Typically used with cylindrical geometry (R-Z) for the Z symmetry axis.</td>
</tr>
<tr>
<td>eqtsymmetric</td>
<td>like reflect for velocities but the magnetic field components, poloidal and toroidal, change sign. The sign of the normal magnetic field component remains the same. Typically used with cylindrical geometry (R-Z) for the R axis to emulate equatorial symmetry.</td>
</tr>
<tr>
<td>hydrostatic-f2</td>
<td>Hydrostatic boundary handling as in FLASH2. See remark in text.</td>
</tr>
<tr>
<td>hydrostatic-f2+nvrefl, hydrostatic-f2+nvout, hydrostatic-f2+nvdiode</td>
<td>Variants of hydrostatic-f2, where the normal velocity is handled specially in various ways, analogous to reflect, outflow, and diode boundary conditions, respectively. See remark in text.</td>
</tr>
<tr>
<td>user-defined user or user</td>
<td>The user must implement the desired boundary behavior; see text.</td>
</tr>
</tbody>
</table>
Flash Code

Parallelisation / Grid-decomposition

• Morton space-filling Z-order decomposition

⟹ results in good load balance

• Note: use of **global** timestep for **all** blocks/refinement levels

⟹ but super-time-stepping (STS) for parabolic equations possible
The directory structure:

```plaintext
Driver/
Grid/
IO/
Multispecies/
Particles/
PhysicalConstants/
RuntimeParameters/
Simulation/
flashUtilities/
monitors/
physics/
```

**e.g.** `Driver_evolFlash.F90`: contains main evolution loop
The directory structure: Grid Unit
The directory structure: Physics Unit

```
Cosmology/
Diffuse/
Eos/
Flame/
Gravity/
Hydro/
RadTrans/
materialProperties/
sourceTerms/
```

- Physics Unit
  - source
    - physics
      - Hydro
        - HydroMain
          - split
            - PPM
              - multiTemp
            - PPMKernel
          - RHD
          - MHD_8Wave
            - Hydro_Unsplit
            - MHD_StaggeredMesh
          - unsplit
FLASH Code

The directory structure: EOS Unit

- physics
  - Eos
    - EosMain
      - Gamma
      - Helmholtz
      - Multigamma
      - multiTemp
      - Tabulated
      - SpeciesBased
      - RHD
The directory structure: source terms

- sourceTerms
  - EnergyDeposition
  - Cool
  - Heat
  - Burn
  - Stir
  - Ionize
  - EnergyDepositionMain
  - BurnMain
  - StirMain
  - IonizeMain

The directory structure: chemistry

- sourceTerms
  - Heatexchange
  - PrimordialChemistry
  - Polytrope
  - HeatexchangeMain
  - PrimordialChemistryMain
  - PrimordialChemistryIntegrate
  - PolytropeMain
  - Constant
  - ConstCoulomb
  - Spitzer
  - GA08
The directory structure: Solvers

- source
- Grid
- GridSolvers
  - Pfft
  - Multipole
  - Multipole_new
  - BHTree
  - Multigrid
  - HYPRE
  - Wunsch
The directory structure:
Lagrangian particles
⇒ e.g. DM particles,
PIC simulations
⇒ see also Particle directory
usage: setup <problem-name> [options] [VAR=VALUE]...

problem-name: see source/Simulation/SimulationMain directory
options:

(Science Options)
- auto [-[123]d]
- maxblocks=<#>  -nxb=<#>  -nyb=<#>  -nzb=<#>
- with-unit=<unit>  -with-library=<libname>[,args]
- without-unit=<unit>  -without-library=<libname>

(Setup and Make Options)
- verbose=[DEBUG|INFO|WARN|IMPINFO|ERROR]
 [-site=<site> | -ostype=<ostype>]
- makefile=<extension>
 [-opt| -debug | -test ]
- objdir=<relative obj directory>
- defines=<defines>  -unitsfile=<filename>
- datafiles=<wildcard>  -parfile=<filename>
- fbs -nofbs -tau=<makefile>
## FLASH Code

### shortcuts for often used setup options

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+cartesian</td>
<td>use cartesian geometry</td>
</tr>
<tr>
<td>+cylindrical</td>
<td>use cylindrical geometry</td>
</tr>
<tr>
<td>+noio</td>
<td>omit IO</td>
</tr>
<tr>
<td>+nolog</td>
<td>omit logging</td>
</tr>
<tr>
<td>+pm2</td>
<td>use the PARAMESH2 grid</td>
</tr>
<tr>
<td>+pm40</td>
<td>use the PARAMESH4.0 grid</td>
</tr>
<tr>
<td>+pm4dev</td>
<td>use the PARAMESH4DEV grid</td>
</tr>
<tr>
<td>+polar</td>
<td>use polar geometry</td>
</tr>
<tr>
<td>+spherical</td>
<td>use spherical geometry</td>
</tr>
<tr>
<td>+ug</td>
<td>use the uniform grid in a fixed block size mode</td>
</tr>
<tr>
<td>+nofbs</td>
<td>use the uniform grid in a non-fixed block size mode</td>
</tr>
<tr>
<td>+usm</td>
<td>use the Unsplit Staggered Mesh MHD solver</td>
</tr>
<tr>
<td>+8wave</td>
<td>use the 8-wave MHD solver</td>
</tr>
<tr>
<td>+unsplitHydro</td>
<td>use the Unsplit Hydro solver</td>
</tr>
</tbody>
</table>
Getting started: run `flash4`

- copy `flash4` and `flash.par` to work directory

- single core run:
  
  ```bash
  ./flash4 [-par_file parameter-filename]
  ```

- parallel run:
  use queuing system: `qsub`, `qstat`, `qdel`, ...

  with q-script
  
  ```bash
  mpirun -n 32 -env I_MPI_FABRICS shm:ofa ./flash4
  ```
Visualisation of simulation results:

- **IDL tools:**
  
  tools/fidlr3.0 $\Rightarrow$ **xflash3 GUI**
  
  $\Rightarrow$ setenv XFLASH3_DIR flash4.0.1/tools/fidlr3.0
  
  $\Rightarrow$ IDL-path:
  
  ```
  !path = !path + '':' + getenv('XFLASH3_DIR')
  ```

- **VisIt**
  
  https://wci.llnl.gov/codes/visit

- **yt:** python based scripts
  
  http://yt-project.org

$\Rightarrow$ all programmes available on **hyades** via module load command
FLASH Code

xflash3 IDL GUI

Prototype File: KH_mhd_2d_hdf5_plt_cnt_0142
suffix: [0142] to [0142] step 1

Output: □ screen □ postscript □ png
Plot size, horizontal: 800 vertical: 500

Parallel Block Distribution: □ Enable Level: 1 □ Show Proc Numbers

Mesh Variables: dens □

Options: □ log □ max □ annotate □ show ticks
□ abs, value □ show blocks □ colorbar

Data range: 0.000 000 to 1.000000e+08 □ auto

Slice Plane: □ x-y □ y-z □ y-z
Zoom: (set = 1 for default)
xrange: [-1] to [-1]
yrange: [-1] to [-1]
zrange: [-1] to [-1]

Contour Options Vector Options Particle Options Histogram Options Floating Label
Plot Histogram Query 1-d Slice

status: awaiting orders . . .
A typical problem setup

• the files in

FLASH4.0.1/source/Simulation/SimulationMain/PROBLEM

define parameters specific to this problem

specific compiler dependencies

declare specific runtime parameters

read in and initialise runtime parameters

set initial grid variables (e.g. density, velocity, ...)

default runtime parameter file for this setup
A typical problem setup: **Simulation_data.F90**

```
module Simulation_data

   implicit none

#include "constants.h"

!!! *** Runtime Parameters *** !!
real, save :: sim_gamma, sim_smallX, sim_smallRho, sim_smallP
real, save :: sim_xMin, sim_xMax, sim_yMin, sim_yMax, sim_zMin, sim_zMax
real, save :: sim_perturbation
logical, save :: sim_gCell, sim_killdivb

integer, save :: sim_meshMe
end module Simulation_data
```

**module**: variables can be accessed within other subroutines
A typical problem setup: \texttt{Simulation_init.F90}

```f90
! FLASH Code

call Driver_getMype(MESH_COMM, sim_meshMe)

call RuntimeParameters_get('gamma', sim_gamma)
call RuntimeParameters_get('xmin', sim_xMin)
call RuntimeParameters_get('ymin', sim_yMin)
call RuntimeParameters_get('zmin', sim_zMin)
call RuntimeParameters_get('xmax', sim_xMax)
call RuntimeParameters_get('ymax', sim_yMax)
call RuntimeParameters_get('zmax', sim_zMax)
call RuntimeParameters_get('killdivb', sim_killdivb)
call RuntimeParameters_get('smallp', sim_smallP)
call RuntimeParameters_get('smallx', sim_smallX)

!! for 3D velocity perturbation
   call RuntimeParameters_get('perturbation', sim_perturbation)
   sim_gCell = .true.

end subroutine Simulation_init
```
FLASH Code

A typical problem setup: the **Config** file

```plaintext
# Configuration file for Orszag Tang MHD vortex problem
# (Orszag and Tang, J. Fluid Mech., 90:129--143, 1979)

REQUIRES physics/Hydro/HydroMain
REQUIRES physics/Eos/EosMain/Gamma

USESETUPVARS withParticles
can be used at setup command

IF withParticles
  PARTICLETYPE passive INITMETHOD lattice MAPMETHOD quadratic ADVMETHOD rungekutta

  REQUIRES Particles/ParticlesMain
  REQUESTS IO/IOMain
  REQUESTS IO/IOParticles
  REQUESTS Particles/ParticlesMapping/Quadratic
  REQUESTS Particles/ParticlesInitialization/Lattice
ENDIF

D tiny Threshold value used for numerical zero
PARAMETER tiny REAL 1.e-16
PARAMETER perturbation REAL 0.2
```

specific runtime parameters with default values
A typical problem setup: the `flash.par` file

```plaintext
# Size of computational volume
xmin = 0.
xmax = 1.
ymin = -0.5
ymax = 0.5

# Boundary conditions
xl_boundary_type = "periodic"
xr_boundary_type = "periodic"
yl_boundary_type = "reflect"
yr_boundary_type = "reflect"

# Simulation (grid, time, I/O) parameters
run_comment = "KH MHD problem"
log_file = "KH_mhd_2d.log"
basenm = "KH_mhd_2d_"
restart = .false.
nend = 10000000
tmax = 10
cfl = 0.8
plot_var_1 = "dens"

#perturbation
perturbation = 0.0125
```
A typical problem setup: \texttt{Simulation_initBlock.F90}

\begin{verbatim}
subroutine Simulation_initBlock(blockID)
  use Simulation_data, ONLY : sim_gCell, sim_gamma, &
      sim_smallX, sim_smallP, &
      sim_killdivb, sim_perturbation
  use Grid_interface, ONLY : Grid_getBlkIndexLimits, &
      Grid_getCellCoords, &
      Grid_getBlkPtr, &
      Grid_releaseBlkPtr
  implicit none

#include "constants.h"
#include "Flash.h"

!!$ Arguments -----------------------------
integer, intent(in) :: blockID
!!$ ----------------------------------------

integer :: i, j, k, n, istat, sizeX, sizeY, sizeZ
real, allocatable, dimension(::) :: xCoord,yCoord,zCoord
integer, dimension(2,MDIM) :: blkLimits, blkLimitsGC
real :: enerZone, ekinZone, eintZone
real, pointer, dimension(:,:,::,:) :: solnData, facexData, faceyData, facezData

call Grid_getBlkIndexLimits(blockID,blkLimits,blkLimitsGC)

sizeX = blkLimitsGC(HIGH,1AXIS)-blkLimitsGC(LOW,1AXIS)+1
sizeY = blkLimitsGC(HIGH,2AXIS)-blkLimitsGC(LOW,2AXIS)+1
sizeZ = blkLimitsGC(HIGH,3AXIS)-blkLimitsGC(LOW,3AXIS)+1
allocate(xCoord(sizeX),stat=istat)
allocate(yCoord(sizeY),stat=istat)
allocate(zCoord(sizeZ),stat=istat)

xCoord = 0.0
yCoord = 0.0
zCoord = 0.0

end subroutine Simulation_initBlock
\end{verbatim}

from \texttt{Simulation_data.F90} \texttt{Simulation_init.F90}
A typical problem setup: set the hydro variables

```fortran
call Grid_getBlkPtr(blockID,solnData,CENTER) ! Loop over cells in the block.
do k = blkLimitsGC(LOW,KAXIS),blkLimitsGC(HIGH,KAXIS)
  do j = blkLimitsGC(LOW,JAXIS),blkLimitsGC(HIGH,JAXIS)
    do i = blkLimitsGC(LOW,IAXIS),blkLimitsGC(HIGH,IAXIS)
      ! Multiple species
      ! solnData(SPECIES_BEGIN,i,j,k)=1.0e0-((NSPECIES-1)*sim_smallX
      do n=SPECIES_BEGIN,SPECIES_END
        solnData(n,i,j,k)=sim_smallX
      enddo
      solnData(PRES_VAR,i,j,k)= sim_pres
      if (yCoord(j) .gt. yMax2) then
        ! Cell-centered values
        solnData(DENS_VAR,i,j,k)= sim_rhoUp
        solnData(VELX_VAR,i,j,k)= sim_Vx0

        solnData(VELY_VAR,i,j,k)= sim_epsilon*sin(sim_modes*2d0*PI*xCoord(i))*exp( -100d0*
        solnData(VELZ_VAR,i,j,k)= 0.
        solnData(TEMP_VAR,i,j,k)= sim_rhoUp*sim_cs*sim_cs/sim_gamma*sim_rhoUp
    enddo
  enddo
enddo
```

access to hydro variables from memory ‘database’
A typical problem setup: set the hydro variables

```
! Compute the gas energy and set the gamma-values needed for the EOS
ekinZone = 0.5 * dot_product(solnData(VELX_VAR:VELZ_VAR,i,j,k),&
                      solnData(VELX_VAR:VELZ_VAR,i,j,k))

! specific internal energy
eintZone = solnData(PRES_VAR,i,j,k)/(sim_gamma-1.)/solnData(DENS_VAR,i,j,k)

! total specific gas energy
enerZone = eintZone + ekinZone

! Take a limit value
enerZone = max(enerZone, sim_smallP)

solnData(ENER_VAR,i,j,k)=enerZone
solnData(EINT_VAR,i,j,k)=eintZone
solnData(GAMC_VAR,i,j,k)=sim_gamma
solnData(GAME_VAR,i,j,k)=sim_gamma
```

make sure the total energy is set properly!
setup your own problem

- create a new directory (e.g. KH) in
  `source/Simulation/SimulationMain/`
- copy setup files from similar setup (e.g. OrszagTang)
- modify setup files
  - Config
  - `Simulation_data.F90`
  - `Simulation_init.F90`
  - `Simulation_initBlock.F90`
  - `flash.par`

- setup and compile
  `. /setup Sod -auto -2d -maxblocks=5000 -objdir=obj_KH`
  `cd obj_KH`
  `make`
- run simulation using new `flash4`
task

try
• Sod-Shock tube problem
• Orszag-Tang MHD vortex problem

• set up a Kelvin-Helmholtz instability problem