Advance Topics in Astrophysical Fluids

Hsi-Yu Schive & Kuo-Chuan Pan

Numerical Astrophysics Summer School 2019: Astrophysical Fluid Dynamics

Multi-dimension

- **Directional splitting:** $\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} = 0 \rightarrow \frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_y}}{\partial y} = 0 \rightarrow \frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0$
  - **Swap directions in the next step to improve accuracy**
    - **E.g.,** $(x, y, z) \rightarrow (z, y, x)$
  - **Pro: more stable (in general), larger timestep** $\Delta t \leq \frac{\Delta h}{max(|v_x| + |v_y| + |v_z|) + C_s}$
  - **Con: break spatial symmetry**
- **Directional unsplitting:** $\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} + \frac{\partial \boldsymbol{F_y}}{\partial y} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0$
  - **Both Riemann solver and data reconstruction are still 1D**
  - **Pro: preserve spatial symmetry**
  - **Con: may be less stable (i.e., negative density/pressure), smaller timestep**
    - **E.g., MUSCL-Hancock:** $\Delta t \leq \frac{\Delta h}{|v_x| + |v_y| + |v_z| + 3C_s}$

Issue of Negative Pressure

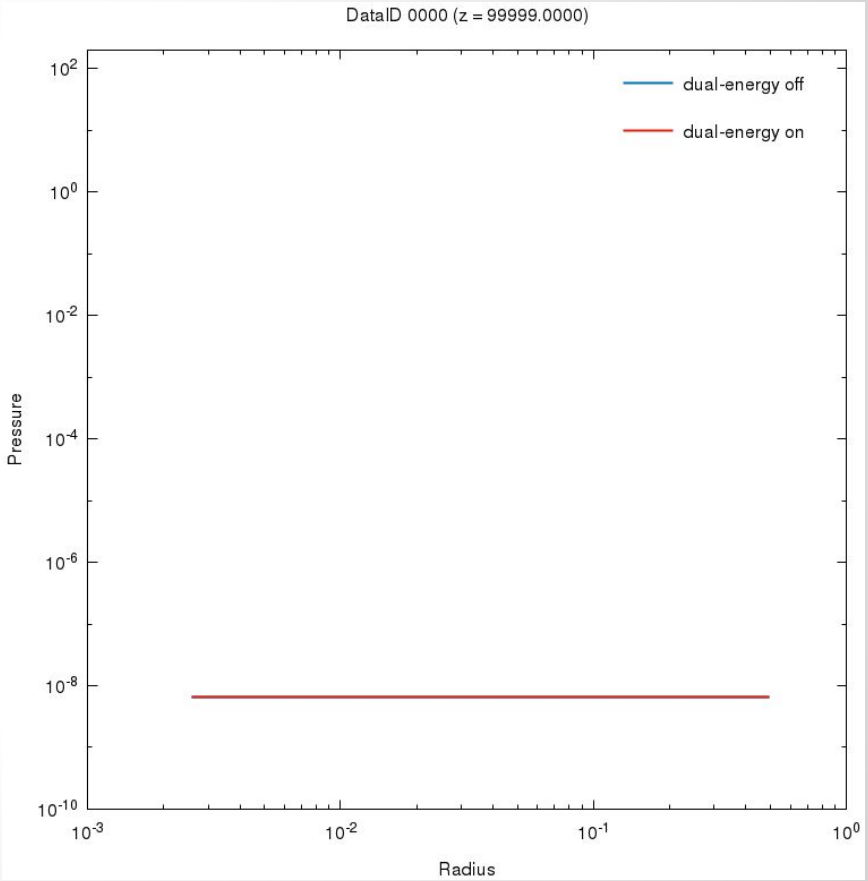
- **Ideal gas:** $P = (\gamma - 1)e = (\gamma - 1)(E - E_k) = (\gamma - 1)\left[E - \frac{(\rho v)^2}{2\rho}\right]$
  - **Conserved variables E, ρ, ρv suffer from truncation errors**
  - **When $E \approx E_k \gg e$ (i.e., high-speed cold flow), e may be seriously contaminated by truncation errors and become negative**
  - **Irrelevant for dynamics, but crucial if temperature is required**
    - **E.g., cooling rate may depend on temperature**
- **Popular solution: dual-energy formalism**
  - **Solve an additional auxiliary eq. → either entropy s or internal energy e**

$$\frac{\partial s}{\partial t} + \boldsymbol{\nabla} \cdot (s\boldsymbol{v}) = 0 \quad \text{or} \quad \frac{\partial e}{\partial t} + \boldsymbol{\nabla} \cdot (e\boldsymbol{v}) = -P\boldsymbol{\nabla} \cdot \boldsymbol{v}$$

  - **Use s or e to compute P only in the regions without shocks**
    - **This is the most tricky part**

Issue of Negative Pressure

- **Example: spherical collapse in cosmology**
- **Solving entropy in this example**
- **Dual-energy formalism significantly improves the solution in the pre-shock (upstream) region**
- **Post-shock (downstream) region is not affected → a strong shock is still captured**



Self-gravity

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0$$

$$\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}\boldsymbol{v} + P\boldsymbol{I}) = -\rho \boldsymbol{\nabla} \phi$$

$$\frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot [(E + P)\boldsymbol{v}] = -\rho \boldsymbol{v} \cdot \boldsymbol{\nabla} \phi$$

Poisson eq.

$$\nabla^2 \phi = 4\pi G \rho$$

- **Operator *splitting* method:**
  - **Procedure**
    - **Solve the original Euler eqs. <u>without</u> gravity**
    - **Solve the Poisson equation for Φ**
    - **Update ρv and E by gravity <u>without</u> considering hydro fluxes**
  - **Disadvantage**
    - **Inadequate for handling hydrostatic equilibrium**
    - **Only 1st-order accurate**

Self-gravity

- Operator *unsplitting* method:
  - Procedure
    - Solve the Poisson equation for ϕ^n
    - Solve the Euler eqs. with the half-step velocity corrected by gravity

      $$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{v} = -\boldsymbol{\nabla} \phi$$

    - Solve the Poisson equation again for ϕ^{n+1}
    - Correct ρv and E with the half-step gravity $\phi^{n+1/2} \approx (\phi^n + \phi^{n+1})/2$
  - Advantage
    - Better for handling hydrostatic equilibrium
    - 2nd-order accurate
- Both methods do NOT conserve total momentum and energy in general
  - Cause: gravity is treated as source terms
  - Solution: rewrite $\rho \boldsymbol{\nabla} \phi$ and $\rho \boldsymbol{v} \cdot \boldsymbol{\nabla} \phi$ into flux-conservative forms

Magnetohydrodynamics (MHD)

- **Ideal MHD:**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \quad \leftarrow \text{mass conservation}$$

$$\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v} - \boldsymbol{B}\boldsymbol{B} + P^* \boldsymbol{I}) = 0 \quad \leftarrow \text{momentum conservation}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P^*)\boldsymbol{v} - \boldsymbol{B}(\boldsymbol{B} \cdot \boldsymbol{v})] = 0 \quad \leftarrow \text{energy conservation}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} - \nabla \times (\boldsymbol{v} \times \boldsymbol{B}) = 0 \quad \leftarrow \text{induction eq. + ideal Ohm's law}$$

- $E = e + \frac{1}{2}\rho v^2 + \frac{B^2}{2}, \quad P^* = P + \frac{B^2}{2}$
- **9 variables to be solved by the 8 equations above + equation of state**
- **Divergence-free constraint on the magnetic field:** $\nabla \cdot \boldsymbol{B} = 0$

Flux-conservative Form for MHD

- $$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} + \frac{\partial \boldsymbol{F_y}}{\partial y} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0,$$

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \\ B_x \\ B_y \\ B_z \end{bmatrix}, \quad \boldsymbol{F_x} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P^* - B_x^2 \\ \rho v_x v_y - B_x B_y \\ \rho v_x v_z - B_x B_z \\ (E + P^*) v_x - B_x (\boldsymbol{B} \cdot \boldsymbol{v}) \\ 0 \\ v_x B_y - v_y B_x \\ v_x B_z - v_z B_x \end{bmatrix}, \text{similarly for } \boldsymbol{F_y}, \boldsymbol{F_z}$$

- **Fluid conserved variables can be updated similarly using the finite-volume scheme for pure hydro**
- **Key question: how to update the magnetic field and ensure the divergence-free constraint?**

Constrained Transport (CT) Method

- **Stokes' theorem:** $$\int_A \frac{\partial \boldsymbol{B}}{\partial t} \cdot d\boldsymbol{A} = \int_A [\boldsymbol{\nabla} \times (\boldsymbol{v} \times \boldsymbol{B})] \cdot d\boldsymbol{A} = \oint_{\partial A} \boldsymbol{v} \times \boldsymbol{B} \cdot d\boldsymbol{l}$$
  - **Electromotive force (EMF):** $\boldsymbol{\varepsilon} = -\boldsymbol{v} \times \boldsymbol{B}$
- **Integrate over cell area (e.g., $\Delta y \Delta z$) and time interval $\Delta t = t^{n+1} - t^n$**

$$B^n_{x,i-1/2,j,k} \equiv \frac{1}{\Delta y \Delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} B_x(x_{i-1/2}, y, z, t^n) dy dz$$

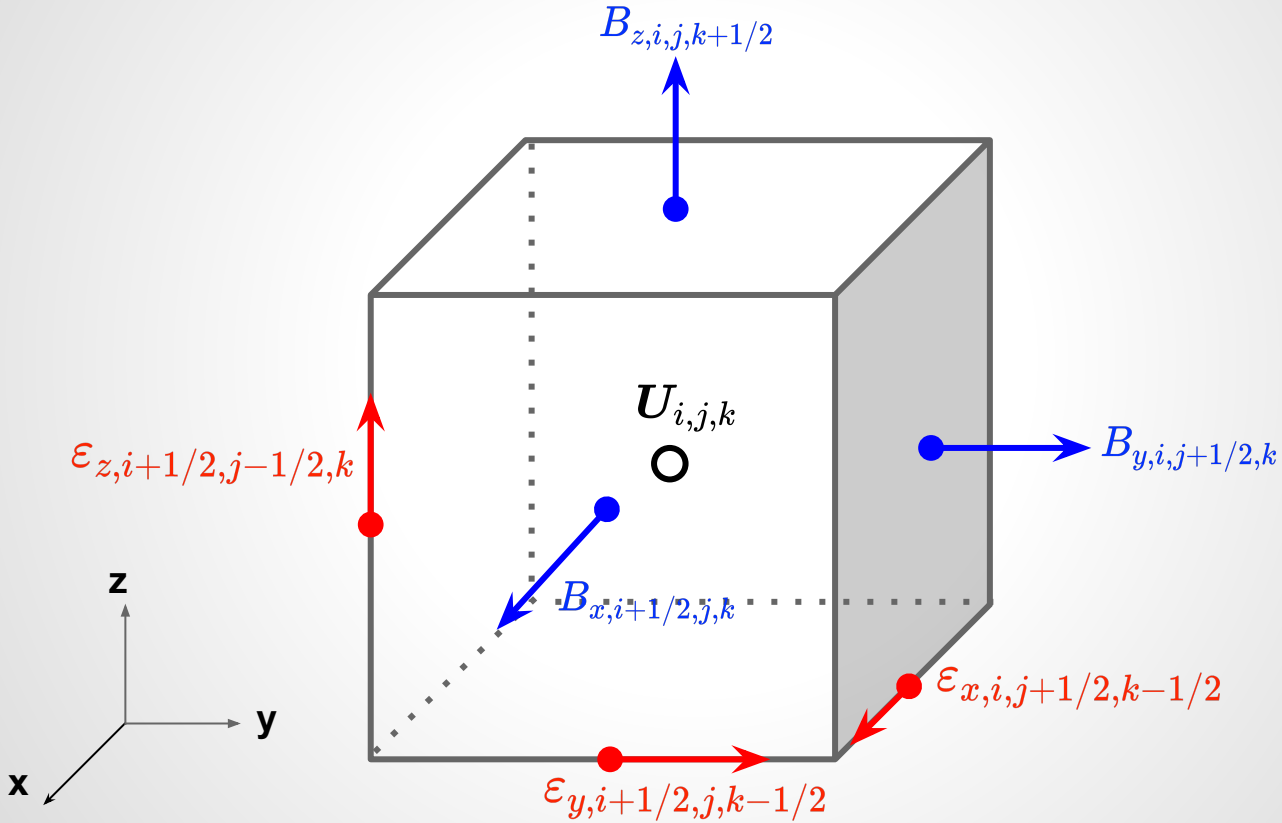
$$\varepsilon^{n+1/2}_{y,i-1/2,j,k-1/2} \equiv \frac{1}{\Delta y \Delta t} \int_{t^n}^{t^{n+1}} \int_{y_{j-1/2}}^{y_{j+1/2}} \varepsilon_y(x_{i-1/2}, y, z_{k-1/2}, t) dy dt$$

$$\varepsilon^{n+1/2}_{z,i-1/2,j-1/2,k} \equiv \frac{1}{\Delta z \Delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \varepsilon_z(x_{i-1/2}, y_{j-1/2}, z, t) dz dt$$

Constrained Transport (CT) Method

- $$B^{n+1}_{x,i-1/2,j,k} = B^{n}_{x,i-1/2,j,k} - \frac{\Delta t}{\Delta y}\left(\varepsilon^{n+1/2}_{z,i-1/2,j+1/2,k} - \varepsilon^{n+1/2}_{z,i-1/2,j-1/2,k}\right) + \frac{\Delta t}{\Delta z}\left(\varepsilon^{n+1/2}_{y,i-1/2,j,k+1/2} - \varepsilon^{n+1/2}_{y,i-1/2,j,k-1/2}\right)$$
  - **This form is again exact → similar to the finite-volume formulation**
  - $B^{n}_{x,i-1/2,j,k}$ **: area-averaged magnetic field**
  - $\varepsilon^{n+1/2}_{z,i-1/2,j\pm1/2,k}$, $\varepsilon^{n+1/2}_{y,i-1/2,j,k\pm1/2}$ **: time- and line-averaged EMF**
- **Similar expressions can be derived for** $B^{n+1}_{y,i,j-1/2,k}$ & $B^{n+1}_{z,i,j,k-1/2}$
- **Area-averaged magnetic field are located at the cell faces instead of centers → staggered grid**

Staggered Grid in CT



Divergence Free in CT

- **Finite-volume representation of the divergence-free constraint:**

$$\frac{1}{\Delta x \Delta y \Delta z} \int_{V_{i,j,k}} (\nabla \cdot \boldsymbol{B^n}) dV = 0$$

exact form

$$\rightarrow \boxed{\begin{aligned}(\nabla \cdot \boldsymbol{B^n})_{i,j,k} = & \frac{B^n_{x,i+1/2,j,k} - B^n_{x,i-1/2,j,k}}{\Delta x} \\ & + \frac{B^n_{y,i,j+1/2,k} - B^n_{y,i,j-1/2,k}}{\Delta y} \\ & + \frac{B^n_{z,i,j,k+1/2} - B^n_{z,i,j,k-1/2}}{\Delta z} = 0\end{aligned}}$$

HW

- **CT update guarantees** $\boxed{\nabla \cdot B^{n+1} = \nabla \cdot B^n}$
  - **Divergence-free constraint is preserved to the machine precision assuming that it is satisfied at the beginning**
  - **The exact way to compute EMF varies from scheme to scheme**

Multi-species

- **In astrophysical environments, it is common that there are more than one species. From species i to N, each species follows,**

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i v) = m_i \Omega_i$$

Ω_i Number density production rate for specie i

$$\sum_i = m_i \Omega_i = 0$$

- **Summing the species equations gives the continuity equation**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0$$

Advanced usage of FLASH code

Advanced usage of FLASH code

- Dimension and Geometry
- Split and Unsplit solvers
- Dual energy
- Self-gravity
- MHD
- Multi-species
- EoS
- Modify your simulation
- Modify AMR rules

Advanced usage of FLASH code

- **Dimension and Geometry**

```
./setup -1d +spherical

./setup -2d +cylindrical

./setup -3d +pm4dev

./setup -3d +ug
```

Table 8.7: Different geometry types. For each geometry/dimensionality combination, the ``support'' column lists the `Grid` implementations that support it: pm4 stands for PARAMESH 4.0 and PARAMESH 4dev, pm2 for PARAMESH 2, UG for Uniform Grid implementations, respectively.

name	dimensions	support	axisymmetric	X -coord	Y -coord	Z -coord
cartesian	1	pm4,pm2,UG	n	x		
cartesian	2	pm4,pm2,UG	n	x	y	
cartesian	3	pm4,pm2,UG	n	x	y	z
cylindrical	1	pm4,UG	y	r		
cylindrical	2	pm4,pm2,UG	y	r	z	
cylindrical	3	pm4,UG	n	r	z	ϕ
spherical	1	pm4,pm2,UG	y	r		
spherical	2	pm4,pm2,UG	y	r	θ	
spherical	3	pm4,pm2,UG	n	r	θ	ϕ
polar	1	pm4,UG	y	r		
polar	2	pm4,pm2,UG	n	r	ϕ	
"polar + z " (cylindrical with a different ordering of coordinates)	3	--	n	r	ϕ	z

Split and unsplit solvers

unsplit (default)

```
./setup +uhd
```

Split solver

```
./setup +splitHydro
```

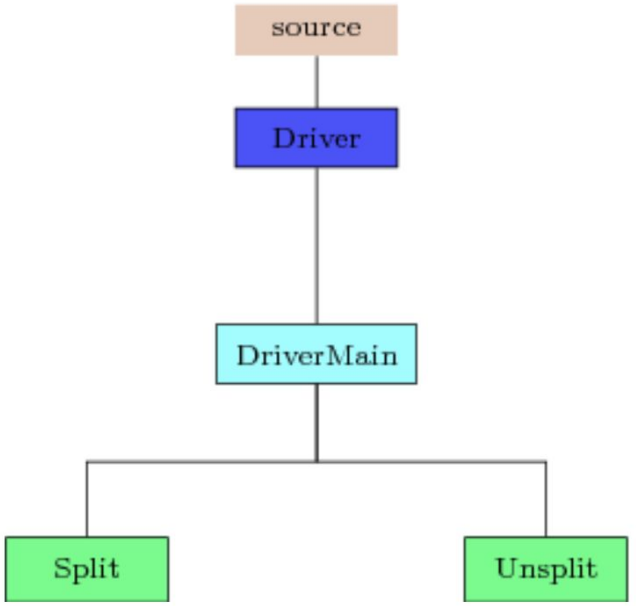


Figure 7.1: The `Driver` unit directory tree.

Runtime parameters for the unsplit Solver

Variable	Type	Default	Description
order	integer	2	Order of method in data reconstruction: 1st order Godunov (FOG), 2nd order MUSCL-Hancock (MH), 3rd order PPM, 5th order WENO.
RiemannSolver	string	“Roe”	Different choices for Riemann solver. ``LLF (local Lax-Friedrichs)'', ``HLL'', ``HLLC'', ``HYBRID'', ``ROE'', and ``Marquina''
use_auxEintEqn	bool	True	Turn on/off solving the auxilary internal energy equation
eintSwitch	real	0	If $\epsilon < \texttt{eintSwitch} \cdot \frac{1}{2}\|\mathbf{v}\|^2$, use the internal energy equation to update the pressure
slopeLimiter	string	“vanLeer”	Slope limiter (mc, vanLeer, vanLeer1.5, minmod, hybrid, limited)

Gravity Units

In setup script,

```
./setup …-unit=source/physics/Gravity/GravityMain/PointMass

./setup …-unit=source/physics/Gravity/GravityMain/Constant

./setup …-unit=source/physics/Gravity/GravityMain/PlanePar
```

In `Config` file,

```
REQUIRES physics/Gravity/GravityMain/PointMass
```

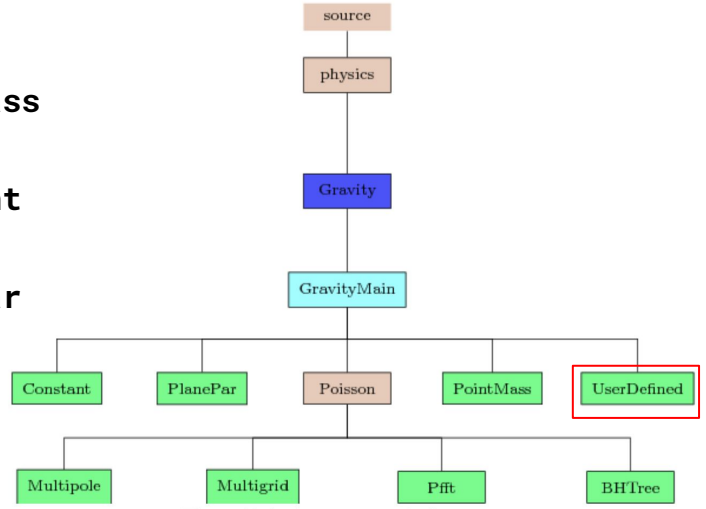


Figure 19.1: The `Gravity` unit directory tree.

Runtime parameters for gravity

Variable	Type	Default	Description
useGravity	Bool	False	Whether gravity calculations should be performed.
grav_boundary_type	String	"isolated"	Type of gravitational boundary condition if a Poisson solve is used for Gravity; string-valued version of grav_boundary. Accepts: "isolated", "periodic", "dirichlet", and maybe others, depending on the Poisson solver used.

Runtime parameters for gravity

```
physics/Gravity/GravityMain/Constant
    gconst [REAL] [-981.]
        Valid Values: Unconstrained
        Gravitational acceleration constant
    gdirec [STRING] ["x"]
        Valid Values: Unconstrained
        Direction of acceleration ("x", "y", "z")
```

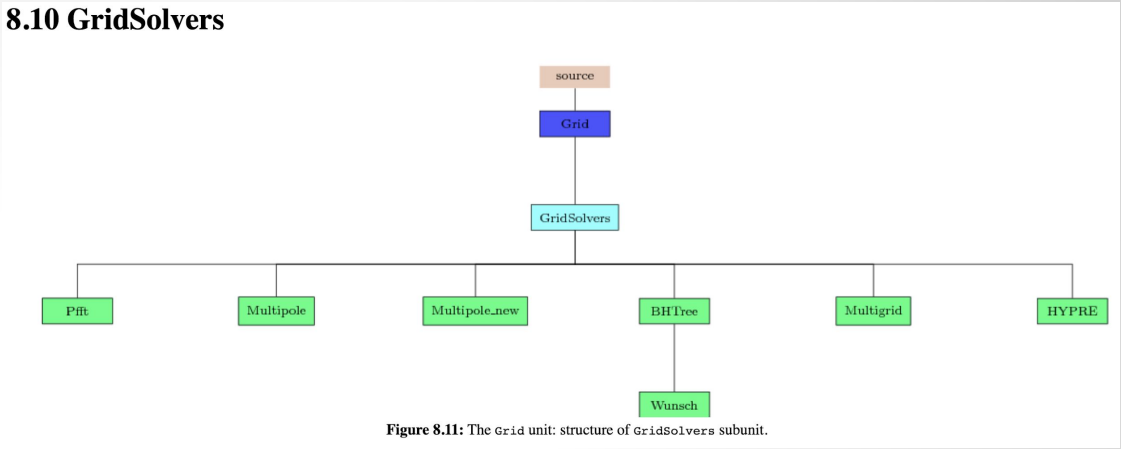
Runtime parameters for gravity

```
physics/Gravity/GravityMain/PlanePar
    gravsoft [REAL] [.0001]
        Valid Values: Unconstrained
        softening length
    ptdirn [INTEGER] [1]
        Valid Values: Unconstrained
        x = 1, y = 2, z = 3
    ptmass [REAL] [10000.]
        Valid Values: Unconstrained
        mass of the point
    ptxpos [REAL] [1.]
        Valid Values: Unconstrained
        location of the point mass, in the ptdirn direction
```

Runtime parameters for gravity

```
physics/Gravity/GravityMain/PointMass
    gravsoft [REAL] [0.001]
        Valid Values: Unconstrained

    ptmass [REAL] [10000.]
        Valid Values: Unconstrained

    ptxpos [REAL] [1.]
        Valid Values: Unconstrained

    ptypos [REAL] [-10.]
        Valid Values: Unconstrained

    ptzpos [REAL] [0.]
        Valid Values: Unconstrained
```

Self-gravity




```
./setup +mpole

./setup +gravmpole

./setup +gravmgrid
```

```
REQUIRES physics/Gravity/GravityMain/Poisson/Multigrid

REQUIRES physics/Gravity/GravityMain/Poisson/Multipole
```

Runtime parameters for gravity

```
physics/Gravity/GravityMain/Poisson
    grav_temporal_extrp [BOOLEAN] [FALSE]
        extrapolate or otherwise rescale
    grav_unjunkPden [BOOLEAN] [TRUE]
        controls whether Gravity_potentialListOfBlocks attempts to restore the
        part of the "pden" ("particle density") UNK variable that is due to
        particles, or leaves "pden" as it is, after a Poisson equation solve.
        This only applies meaningfully when a "pden" variable is declared and
        the gravitational potential is calculated by solving a Poisson equation
        whose right-hand side includes a mass distribution to which both
        hydrodynamic fluid density and massive particles contribute. The "pden"
        variable will have been set to the sum of the fluid density ("dens"
        variable) and the density resulting from mapping massive particles to
        the mesh, so that is what remains in "pden" when grav_unjunkPden is set
        to FALSE. Otherwise, "dens" will be subtraced from "pden" before
        Gravity_potentialListOfBlocks returns, and "pden" will be left
        containing only the mass density that is due to particles.
    point_mass [REAL] [0.e0]
        Valid Values: Unconstrained
        mass of the central point-like object
    point_mass_rsoft [REAL] [0.e0]
        Valid Values: Unconstrained
        softening radius for the point-like mass (in units of number of the
        finest level cells)
    updateGravity [BOOLEAN] [TRUE]
        allow gravity value to be updated
```

Equation of States

- **Gamma-law equation of state could be invalid in some astrophysical environments in which electrons and positrons may be relativistic and/or degenerate, and in which radiation may significantly contribute to the thermodynamic state.**
- **The Helmholtz EOS includes contributions from radiation, completely ionized nuclei, and degenerate/relativistic electrons and positrons. The pressure and internal energy are calculated as the sum over the components.**

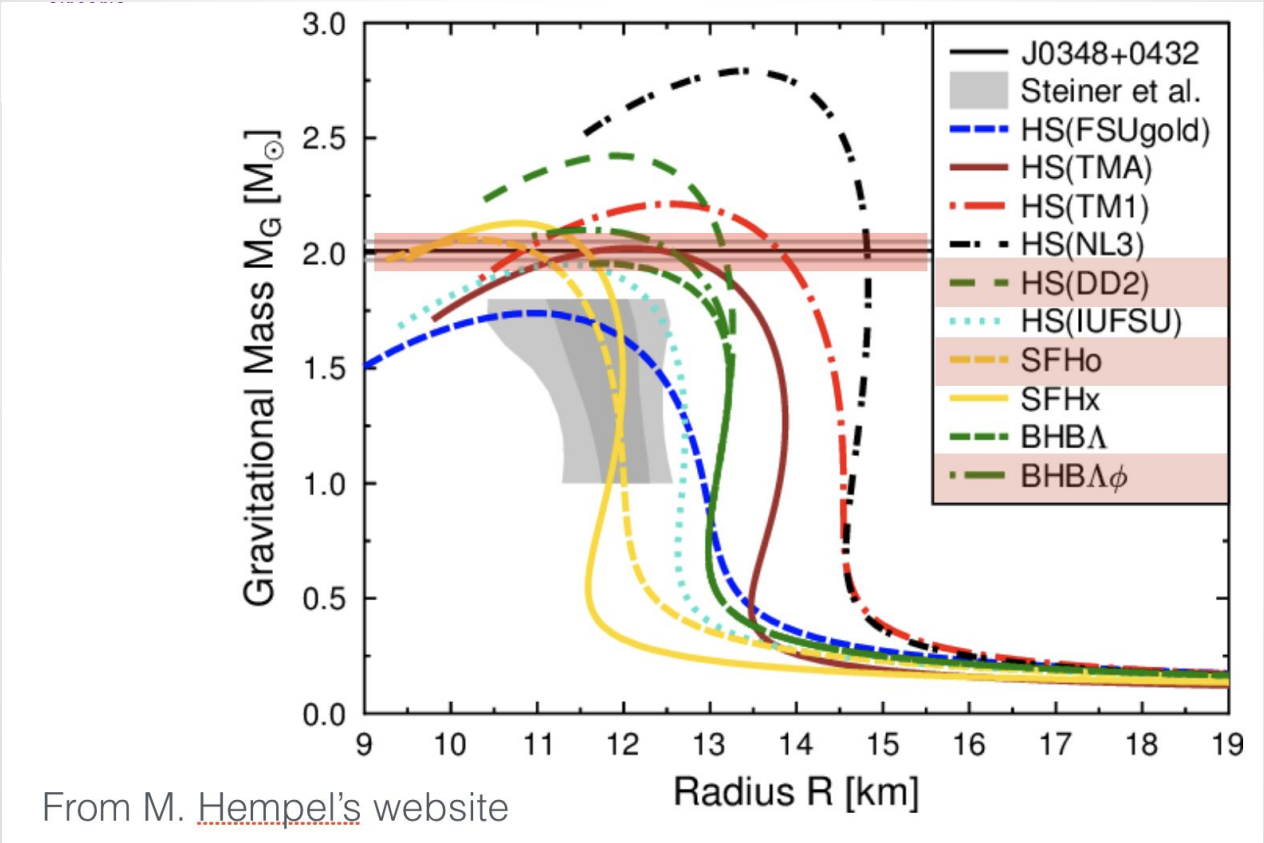
$$P_{\rm tot} = P_{\rm rad} + P_{\rm ion} + P_{\rm ele} + P_{\rm pos} + P_{\rm coul}$$

Equation of States

- **Extreme hot and dense environments can be realized in core-collapse supernovae, which occur at the very end of massive stellar evolutions and lead to the formation of a neutron star or a black hole.**
- **At high temperature $T > 0.4\,\mathrm{MeV}$, chemical equilibrium is achieved for all strong and electromagnetic reactions, which is referred to as nuclear statistical equilibrium (NSE) and the nuclear composition is determined as a function of density, temperature, and proton (electron) fraction.**

$$10^5 < \rho < 10^{15}\ \mathrm{g/cm^3} \qquad 0.1 < T < 100\ \mathrm{MeV}$$

Nuclear EoS



From M. Hempel's website

Multi-species

source/Simulation/SimulationComposition/Burn

```
 1 neut    0   1   1   0   2   0.5
 2 h1      1       1       0       0       2       0.5
 3 prot    1   1   0       0   2   0.5
 4 he3 2   3   1   7.718   2   0
 5 he4 2   4   2       28.296  1   0
 6 c12 6   12  6       92.162  1   0
 7 n14     7       14      7       104.659 1       0
 8 o16 8   16  8       127.619 1   0
 9 ne20    10  20  10      160.645 1   0
10 ne21    10  21  11      167.406 4   0
11 ne22    10  22  12      177.77  1   0
12 na23    11  23  12      186.564 4   0
13 mg23    12  23  11      181.726 4   0
14 mg24    12  24  12      198.257 1   0
15 mg25    12  25  13      205.588 6   0
16 mg26    12  26  14      216.681 1   0
17 al25    13  25  12      200.529 6   0
18 al26    13  26  13      211.894 11  0
```

< Example: species in Burn >

Modify your simulation

Simulation: Simulation_adjustSimulation(blkcnt, blklst, nstep, dt, stime)

Physics

Heating: Heat(blockCount,blockList,dt,time)

Cooling: Cool(blockCount,blockList,dt,time)

< Example: nuclear heating in Project A >

Modify AMR

API: Grid_markRefineDerefine(), Grid_markRefineSpecialized

SYNOPSIS

```
  Grid_markRefineSpecialized(integer(IN) :: criterion,
                             integer(IN) :: size,
                             real(IN)    :: specs(size),
                             integer(IN) :: lref )
```

Modify AMR

DESCRIPTION

The routine provides an interface to a collection of routines that define very specialized refinement criteria. The currently supported options are:

THRESHOLD	: when a specific variable is below or above a threshold
ELLIPSOID	: The blocks that fall within the specified ellipsoid
RECTANGLE	: The blocks that fall within the specified rectangle
INRADIUS	: The blocks that fall within the specified radius
WITHRADIUS	: The blocks that fall on the specified radius

Modify AMR

```
ARGUMENTS
  criterion - the creterion on which to refine
  size      - size of the specs data structure
  specs     - the data structure containing information specific to
              the creterion
              For THRESHOLD
                 specs(1) = real(variable_name), for example
                           if variable is density, then
                           specs(1)=real(DENS_VAR)
                 specs(2) = the threshold value
                 specs(3) = if < 0 refine if variable < threshold
                            if > 0 refine if variable > threshold
              For ELLIPSOID
                 specs(1:3) = center of the ellipsoid
                 specs(4:6) = the semimajor axes of the ellipsoid
              For INRADIUS
                 specs(1:3) = center of the circle/sphere
                 specs(4)   = the radius
              For WITHRADIUS
                 specs(1:3) = center of the circle/sphere
                 specs(4)   = the radius
              For RECTANGLE
                 specs(1:6) = bounding coordinates of rectangle
                 specs(7)   = if 0 refine block with any overlap
                              if /= refine only blocks fully
                              contained in the rectangle


  lref      - If > 0, bring selected blocks to this level of refinement.
              If <= 0, refine qualifying blocks once.
```