

The EAGLE code

Chapter 1

TO do's

- re-make plots of EOS
- compare kinetic feedback rob versus me
- change name of eos flag in sfr
- change temperature limit for sfr
- insert stellar evolution
- insert vel disp calculation
- add temperature floor to eagle_sfr, below which stars are allowed to form (in case they are not on the eos yet)
- add new parameter: minimum reheating temperature for type II SNe
- add new parameter: minimum re-heating temperature for type I SNe and implement type I SNe re-heating in similar way as type II. For that you need to compute the ngb_mass weight, also in eagle_enrich
- star's smoothing lengths are not read from a snapshot file -> change this

Chapter 2

Issue's

- routine `eagle_timestep.c`: why is entropy *derivative* zeroed if predicted entropy \leq floor? Should it not be that entropy derivative cannot be negative?
- routine `eagle_stellar_evolution`, line 232: number of SNe is calculated by multiplying specific number by (initial) mass of star – there seems to be a $1/h$ missing, since gadget masses are in h^{-1} mass units (cfr similar calculation in routine `eagle_stellar_feedback`, line 172).

Chapter 3

The sub-grid model in EAGLE

3.1 Hydro and gravity

This section explains the equations that are being integrated, with a focus on definitions of variables and their dependence on a (the expansion factor).

The equations we need to solve, those of a self-gravitating fluid, are the continuity, Euler, energy and Poisson equations. They are respectively

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \mathbf{v}) = 0 \quad (3.1)$$

$$\frac{\partial}{\partial t} \mathbf{v} + (\mathbf{v} \cdot \nabla) \mathbf{v} = -\frac{1}{\rho} \nabla p - \nabla \Phi \quad (3.2)$$

$$\rho \frac{\partial}{\partial t} u + \rho (\mathbf{v} \cdot \nabla) u = -p \nabla \mathbf{v} \quad (3.3)$$

$$\nabla^2 \Phi = 4\pi G \rho. \quad (3.4)$$

Here, u is the energy per unit mass, $u = p/(\gamma - 1)\rho = k_B T/(\gamma - 1)\mu m_h$.

To take into account the expansion of the Universe, we will write the physical variables \mathbf{r} (the position) and \mathbf{v} (velocity) as perturbations on top of a homogeneous expansion, by introducing the *co-moving position* \mathbf{x} and the *peculiar velocity* \mathbf{v}_p as

$$\begin{aligned} \mathbf{r} &= a(t)\mathbf{x} \\ \mathbf{v} = \dot{\mathbf{r}} &= \dot{a}\mathbf{x} + a\dot{\mathbf{x}} \equiv \dot{a}\mathbf{x} + \mathbf{v}_p. \end{aligned} \quad (3.5)$$

Here, $a(t)$ is the scale factor of the expanding (background) cosmological model. The velocity \mathbf{v} is the sum of the Hubble velocity, $\dot{a}\mathbf{x} = H\mathbf{r}$, where $H = \dot{a}/a$ is Hubble's constant, and a 'peculiar' velocity, $\mathbf{v}_p = a\dot{\mathbf{x}}$. In what follows we will write co-moving variables with a hat to distinguish them from physical variables. So $\hat{r} = a^{-1}r = x$, and also $\hat{\nabla} \equiv \partial/\partial x = a\nabla$.

The Euler equation written in terms of x and \dot{x} is

$$a\ddot{x} + 2a\dot{a}\dot{x} + \ddot{a}x = -\frac{1}{\rho}\nabla p - \nabla\Phi. \quad (3.6)$$

The $\ddot{a}x$ term drops-out by changing the potential from Φ to Ψ , defined as

$$\Phi = \Psi + \frac{2\pi}{3}G\rho_b r^2 - \frac{1}{6}\Lambda r^2 \quad (3.7)$$

$$\nabla^2\Phi = 4\pi G\rho_b r^2\delta, \quad (3.8)$$

where δ is now the over density, and ρ_b the mean density.

In Gadget II (and Gadget III), position and velocity¹ are

$$\text{Pos} \equiv x \quad (3.9)$$

$$\text{Vel} \equiv a^2 \frac{dx}{dt} \quad (3.10)$$

The Gadget expressions for the equation of motion are then

$$\frac{d\text{Pos}}{dt} = \frac{1}{a^2} \text{Vel} \quad (3.11)$$

$$\frac{d\text{Vel}}{dt} = -a \sum \frac{Gm}{r^2} - a \frac{\nabla p}{\rho}, \quad (3.12)$$

where r is still the *physical* position, and $\nabla \equiv \partial r$.

Time variables

Gadget uses the expansion factor a as time variable, $d \log a$ as the time step, and uses integer stepping to get accurate times. The variables involved are

¹Note this is true inside the code, see below for the definitions of variables stored in snapshots.

$$\text{Timebase_interval} \equiv \frac{\log \text{TimeMax}/\text{TimeBegin}}{\text{TIMEBASE}} \quad (3.13)$$

$$dt_{\text{phys}} = \frac{\text{smoothinglength}}{c} \quad (3.14)$$

$$dt_{\text{step}} = dt_{\text{phys}} \times \frac{\dot{a}}{a} = d \log a \quad (3.15)$$

$$ti_{\text{step}} = \frac{d \log a}{\text{Timebase_interval}}. \quad (3.16)$$

Here, TIMEBASE devides the logarithm of the ratio of $a_{\text{begin}} = \text{TimeBegin}$ over $a_{\text{max}} = \text{TimeMax}$, i.e. the value of the expansion factor at the end of the simulation over its value at the start, in TIMEBASE equal steps, where TIMEBASE is a power of two. dt_{phys} is an example of a physical time step (here the Courant step), this is multiplied by \dot{a}/da (the Hubble constant), to obtain $dt_{\text{step}}=d \log(a)$, the log of the change in expansion factor. Finally, ti_{step} is the integer time step that then corresponds to this physical time step. The value of ti_{step} is decreased until it is a power of two. The timestep bin associated with this step is also recorded, ie $2^{\text{bin}}=ti_{\text{step}}$.

Some useful conversions are then:

get time step from bin:

$$d \log a = 2^{\text{bin}} \times \text{Timebase_interval} \quad (3.17)$$

$$dt = \frac{d \log a}{h} \quad (3.18)$$

get time step from a difference between two gadget time, All.Time and $dt = d \log a$

Some often used definitions

$$\text{hubble} \equiv \frac{\dot{a}}{a} \quad (3.19)$$

$$(3.20)$$

Update of positions

In the PREDICT routine, the Pos variables are updated as

$$\Delta \text{Pos} = \text{Vel} \frac{dt}{a^2} = \text{Vel} \int_{a_1}^{a_2} \frac{da}{a^3 h(a)} = \text{Vel} dt_{\text{drift}} \quad (3.21)$$

$$dt_{\text{drift}} \equiv \int_{a_1}^{a_2} \frac{da}{a^3 h(a)} \quad (3.22)$$

The time-step dt_{drift} is interpolated from a pre-computed table in routine DRIFTFAC.C.

Gravity.

The gravitational forces returned by the PM and tree-code are

$$\text{Acc}_g = \sum \frac{G m}{\text{Pos}^2} . \quad (3.23)$$

The update of velocities when kicked, in terms of the gravity acceleration Acc_g , are thus

$$\Delta \text{Vel} = \frac{1}{a} \text{Acc}_g dt = \text{Acc}_g \int_{a_1}^{a_2} \frac{da}{a^2 h} = \text{Acc}_g dt_{\text{grav_kick}} \quad (3.24)$$

$$dt_{\text{grav_kick}} \equiv \int_{a_1}^{a_2} \frac{da}{a^2 h} \quad (3.25)$$

The time-step $dt_{\text{grav_kick}}$ is interpolated from a pre-computed table in routine DRIFTFAC.C.

Hydro

The co-moving density, pressure, thermal energy, and entropy are related to their physical values through

$$\hat{\rho} = a^3 \rho \quad (3.26)$$

$$\hat{p} = a^{3\gamma} p \quad (3.27)$$

$$\hat{u} = a^{3(\gamma-1)} u \quad (3.28)$$

$$\hat{S} = S \quad (3.29)$$

$$\hat{c} = a^{3(\gamma-1)/2} c , \quad (3.30)$$

where $c^2 = \gamma p / \rho$ is the sound speed. Note that these definitions are somewhat arbitrary and other codes use other definitions. In particular these definitions have the disadvantage that the velocity variable Vel , and the co-moving sound-speed \hat{c} , have different dependencies on the expansion factor a , hence for example the sum of velocity and sound-speed (as may appear in the calculation of a signal velocity) should be performed as

$$a\dot{x} + c = \frac{1}{a} \text{Vel} + a^{-3(\gamma-1)/2} \hat{c} = \frac{1}{a} [\text{Vel} + a^{1-3(\gamma-1)/2} \hat{c}] . \quad (3.31)$$

Note further that $\hat{p} = (\gamma - 1) \hat{\rho} \hat{u} = \hat{S} \hat{\rho}^\gamma$, therefore the equation of state does not change, and in particular $\hat{c}^2 = \gamma \hat{p} / \hat{\rho}$.

Changing to co-moving variables in this way, we find that

$$\frac{\nabla p}{\rho} = \frac{1}{a a^{3(\gamma-1)}} \frac{\hat{\nabla} \hat{p}}{\hat{\rho}} \quad (3.32)$$

$$\frac{d\text{Vel}}{dt} = -a \frac{\nabla p}{\rho} = -\frac{1}{a^{3(\gamma-1)}} \frac{\hat{\nabla} \hat{p}}{\hat{\rho}} \quad (3.33)$$

The hydro-acceleration returned by routine `hydra` are

$$\text{Acc}_h = -\frac{\hat{\nabla} \hat{p}}{\hat{\rho}}, \quad (3.34)$$

and therefore the update of `Vel` due to hydro accelerations is

$$\begin{aligned} \Delta \text{Vel} &= \text{Acc}_h \frac{dt}{a^{3(\gamma-1)}} = \text{Acc}_h \int_{a_1}^{a_2} \frac{1}{a h(a) a^{3(\gamma-1)}} da \\ &= \text{Acc}_h dt_{\text{hydro_kick}} \end{aligned} \quad (3.35)$$

$$dt_{\text{hydro_kick}} \equiv \int_{a_1}^{a_2} \frac{1}{a h(a) a^{3(\gamma-1)}} da. \quad (3.36)$$

The time-step $dt_{\text{hydro_kick}}$ is interpolated from a pre-computed table in routine `DRIFTFAC.C`.

The hydro accelerations as implemented in `Gadget2` follow from

$$\nabla \frac{p}{\rho} = \frac{\nabla p}{\rho} - \frac{p}{\rho^2} \nabla \rho, \quad (3.37)$$

and the usual SPH way of finding smoothed quantities, such as

$$\rho(i) = \sum_j m_j W_{ij} \quad (3.38)$$

$$A_i \rho_i = \sum_j m_j A_j W_{ij}, \quad (3.39)$$

in terms of the kernel W_{ij} and the sum over neighbours j . Therefore

$$\frac{dv}{dt} = -\frac{\nabla p}{\rho}_i = -\sum_j m_j \left[\frac{p_i}{\rho_i^2} \nabla W_i + \frac{p_j}{\rho_j^2} \nabla W_j \right] \quad (3.40)$$

where W_i uses the smoothing length h_i and vice versa for j . This is Eq. (7) in the Gadget 2 paper (apart from the f_i factors). The artificial viscosity term used in Gadget 2 adds another term to the acceleration, which is

$$\frac{dv}{dt}|_{\text{visc}} = - \sum_j m_j \Pi_{ij} \nabla W \quad (3.41)$$

$$\Pi_{ij} = -\frac{\alpha}{2} \frac{(c_i + c_j - 3w_{ij}) w_{ij}}{\rho_{ij}} \quad (3.42)$$

$$w_{ij} = \frac{v_{ij} \cdot r_{ij}}{r_{ij}}. \quad (3.43)$$

These are equations (9) and (14) of the gadget 2 paper. I next describe how these equations are implemented, following the naming convention of Gadget 3, by introducing variables `fac_mu`, `vdotr2`, `mu_ij`

Note first that the kernel and its spatial derivative transform as

$$\hat{W} = a^3 W \quad (3.44)$$

$$\hat{\nabla} \hat{W} = a^4 \nabla W. \quad (3.45)$$

For a spherically symmetric kernel, gradients are

$$\nabla W = \mathbf{r} \frac{1}{r} \frac{\partial W}{\partial r}. \quad (3.46)$$

Next note that the relative radial velocity

$$v \cdot r = (a\dot{x} + \dot{a}x) \cdot ax = a^2 \dot{x} \cdot x + a^2 h\text{Pos} \equiv \text{vdotr2}. \quad (3.47)$$

The following gadget variables are then

$$\text{fac_mu} = a^{-1} a^{3(\gamma-1)/2} \quad (3.48)$$

$$\text{mu_ij} = \frac{\text{fac_mu} \text{vdotr2}}{\text{Pos}} \quad (3.49)$$

$$= a^{3(\gamma-1)/2} \frac{v \cdot r}{r} \quad (3.50)$$

$$= a^{3(\gamma-1)/2} w_{ij} \quad (3.51)$$

$$\text{v_sig} = (\hat{c}_i + \hat{c}_j - 3\text{mu_ij}) \quad (3.52)$$

$$= a^{3(\gamma-1)/2} (c_i + c_j - 3w_{ij}) \quad (3.53)$$

$$\text{visc} = \frac{\alpha \text{vsig} (-\text{mu_ij})}{2 \hat{\rho}} \quad (3.54)$$

$$= -\frac{a^{3(\gamma-1)}}{a^3} \frac{\alpha (c_i + c_j - 3w_{ij}) w_{ij}}{2 \rho_{ij}} \quad (3.55)$$

$$= \frac{a^{3(\gamma-1)}}{a^3} \Pi_{ij} \quad (3.56)$$

$$\text{dwk_i} = \frac{1}{h^4} \frac{1}{q} \frac{\partial W(q)}{\partial q} \quad (3.57)$$

$$\text{hfc} = m_j (\text{visc} + (\frac{\hat{p}_i}{\hat{\rho}_i^2} + \frac{\hat{p}_j}{\hat{\rho}_j^2}) \frac{\text{dwk_ij}}{\text{Pos}}) \quad (3.58)$$

$$= \frac{a^{3\gamma}}{a^2} \left[\Pi_{ij} + \frac{p}{\rho^2} \right] \frac{1}{r} \frac{\partial W}{\partial r} \frac{1}{\text{Pos}}. \quad (3.59)$$

Note that in the last line $\text{dwk_i} = a^4 r^{-1} \partial W / \partial r$. Finally the hydro acceleration is

$$\text{Acc}_{h,x} = -\text{hfc} \Delta \text{Pos_x} \quad (3.60)$$

$$= -a a^{3(\gamma-1)} \left[\Pi_{ij} + \frac{p}{\rho^2} \right] \frac{1}{r} \frac{\partial W}{\partial r} \frac{\Delta \text{Pos_x}}{\text{Pos}} \quad (3.61)$$

$$= -a a^{3(\gamma-1)} \frac{\nabla p}{\rho} \frac{\Delta \text{Pos_x}}{\text{Pos}} \quad (3.62)$$

$$= -\frac{\hat{\nabla} \hat{p}}{\hat{\rho}} \frac{\Delta \text{Pos_x}}{\text{Pos}} \quad (3.63)$$

as should be.

Entropy

Entropy, $A = p/\rho^\gamma$, changes due to the action of the artificial viscosity term changes the entropy at a rate (Eq. 10 in the GADGET paper)

$$\frac{dA}{dt} = \frac{1}{2} \frac{\gamma - 1}{\rho^{\gamma-1}} \sum m \Pi_{ij} v_{ij} \cdot \nabla W. \quad (3.64)$$

The follows from energy conservation, where the change in thermal energy, $u = \rho^{(\gamma-1)} A / (\gamma-1)$ is due to viscous work done, $du/dt = -(d\mathbf{v}/dt)_{\text{visc}} \cdot \mathbf{v}$. The factor 1/2 is due to the symmetrization of the velocity, $\mathbf{v} \equiv \mathbf{v}_j - \mathbf{v}_i$.

Note first that

$$\mathbf{v} \cdot \nabla W = v_x \frac{x}{r} \frac{\partial W}{\partial r} + \dots \quad (3.65)$$

$$= (\mathbf{v} \cdot \mathbf{r}) \frac{1}{r} \frac{\partial W}{\partial r}. \quad (3.66)$$

Therefore

$$\frac{d\hat{A}}{dt} = \frac{dA}{dt} \quad (3.67)$$

$$= \frac{1}{2} \frac{\gamma - 1}{a^{-3(\gamma-1)}} \sum m_j \frac{a^3}{a^{3(\gamma-1)}} \text{visc} \cdot \mathbf{v} \cdot \mathbf{r} \frac{1}{a^5} \frac{1}{\hat{r}} \frac{\partial \hat{W}}{\partial \hat{r}} \quad (3.68)$$

$$= \frac{1}{a^2} \frac{1}{2} \frac{\gamma - 1}{\hat{\rho}^{\gamma-1}} \sum \text{hfc_visc} \cdot \mathbf{v} \cdot \mathbf{r}. \quad (3.69)$$

The rate of change of entropy is stored in the variable

$$\text{dtEntropy} = \frac{1}{a^2 h} \frac{\gamma - 1}{2} \frac{1}{\hat{\rho}^{\gamma-1}} \sum_j \text{hfc_visc} \cdot \mathbf{v} \cdot \mathbf{r}. \quad (3.70)$$

$$= \frac{1}{h} \frac{d\hat{A}}{dt}, \quad (3.71)$$

where the multiplication by $(1/(a^2 h)) ((\gamma - 1)/2) (1/\rho^{\gamma-1})$ is performed outside the loop over j . Therefore \hat{A} is updated as

$$\Delta \hat{A} = \frac{d\hat{A}}{dt} dt \quad (3.72)$$

$$= \frac{1}{\dot{a}} \frac{d\hat{A}}{dt} da \quad (3.73)$$

$$= \text{dtEntropy} \frac{da}{a} \quad (3.74)$$

$$= \text{dtEntropy} \text{ dt_step}. \quad (3.75)$$

Note in particular that indeed $dt_step = da/a$ as explained before.

Curl and divergence

The density routine calculates curls and gradients as follows. First note that

$$\nabla \rho \mathbf{v} = \nabla \rho \cdot \mathbf{v} + \rho \nabla \cdot \mathbf{v} \quad (3.76)$$

$$\nabla \cdot \mathbf{v} = \frac{1}{\rho} (\nabla \rho \mathbf{v} - \nabla \rho \cdot \mathbf{v}) \quad (3.77)$$

$$(\nabla \cdot \mathbf{v})_i = \frac{1}{\rho_i} \sum m_j (\mathbf{v} \cdot \mathbf{r}) \frac{1}{r} \frac{\partial W}{\partial r}, \quad (3.78)$$

where the last line is the corresponding SPH expression.

Note that in an expanding Universe, where the peculiar velocity $\mathbf{v}_p \equiv a\dot{\mathbf{x}}$,

$$\nabla \cdot \mathbf{v} = 3\frac{\dot{a}}{a} + a\nabla \cdot \dot{\mathbf{x}} = 3h + \nabla \cdot \mathbf{v}_p. \quad (3.79)$$

Note that $\text{Pos} \cdot \text{Vel} = \mathbf{x} \cdot a^2\dot{\mathbf{x}} = \mathbf{r} \cdot \mathbf{v}_p$, therefore the divergence of the *peculiar* velocity in SPH is

$$(\nabla \cdot \mathbf{v})_p = \frac{1}{a^{-3}\hat{\rho}_i} \sum_j m_j (\text{Pos} \cdot \text{Vel}) \frac{1}{a^5} \frac{1}{\hat{r}} \frac{\partial \hat{W}}{\partial \hat{r}} \quad (3.80)$$

$$= \frac{1}{a^2} \text{DivVel}. \quad (3.81)$$

The last line introduces the Gadget variable DivVel. I see no reason why the term $3h$ is missing from the calculation of the divergence.

Update of smoothing lengths

The smoothing length and the density are obtained from solving the following intrinsic equation

$$\frac{4\pi}{3} \rho h^3 = N_{\text{SPH}} \bar{m}. \quad (3.82)$$

If we have a guess h_0 of the smoothing length, which gives a current particle weight N_0 , and a density ρ_0 , we can try to improve the guess by starting from the equation for the SPH density,

$$\rho = \sum_j m_j \frac{1}{h^3} W(q), \quad (3.83)$$

where $q = r/h$. For this, the derivative

$$\frac{\partial \rho}{\partial h} = \sum_j m_j \left[-\frac{3}{h^4} W - \frac{q}{h^4} \frac{\partial W}{\partial q} \right] \quad (3.84)$$

$$= - \sum_j m_j \left[\frac{3}{h} \text{kernel.Wk} + \frac{q}{h} \text{kernel.dWk} \right]. \quad (3.85)$$

This quantity is calculated in the density loop, together with ρ and N_0 (the current particle weight). Let the desired smoothing length be h_i , and the corresponding density ρ_i , then $\rho_i = \rho_0 + (\partial \rho / \partial h) (h_i - h_0)$, and, since approximately $\rho_i h_i^3 / N_i = \rho_0 h_0^3 / N_0$, we get

$$\rho_0 h_0^3 = \frac{N_0}{N_i} h_i^3 \quad (3.86)$$

$$= \frac{N_0}{N_i} h_i^3 \left[\rho_0 + \frac{\partial \rho}{\partial h} (h_i - h_0) \right] \quad (3.87)$$

$$= \frac{N_0}{N_i} [h_0 + \Delta h]^3 + \left[\rho_0 + \frac{\partial \rho}{\partial h} \Delta h \right] \quad (3.88)$$

$$\approx \frac{N_0}{N_i} \rho_0 h_0^3 \left[1 + \frac{3\Delta h}{h_0} \right] \left[1 + \frac{\partial \rho}{\partial h} \frac{\Delta h}{\rho_0} \right], \quad (3.89)$$

where $\Delta h \equiv h_i - h_0$, or, in terms of $\epsilon \equiv \Delta h / h_0$,

$$\epsilon = \frac{N_i - N_0}{N_0} \frac{1}{3 + \frac{h_0}{\rho_0} \frac{\partial \rho}{\partial h}} \quad (3.90)$$

$$h_i = h_0 \left(1 - \frac{N_0 - N_i}{3N_0} \right) \frac{1}{1 + \frac{h_0}{3\rho_0} \frac{\partial \rho}{\partial h}}. \quad (3.91)$$

Note that the "3" is actually the number of dimensions. This way of updating h is used in the density calculation, and it improves the rate of convergence especially when close to the right answer. For some reason, it is not used in the calculation of densities (or smoothing lengths) for stars or black holes.

3.1.1 Time steps

The time step for the particles is compute in physical units and then converted to comoving $d \log(a)$ values. This conversion is described at the beginning of section 3.1. This subsection describes the time steps used by the

gas particles in physical units.

On top of the usual criterion for gravity acceleration and time steps, the code uses the Courant condition. For each particle, the signal velocity $v_{sig,i}$ is computed in the density loop. The time step size is then given by the CFL condition on the resolution element of size h_i :

$$\Delta t_i = 2C_{CFL} \frac{h_i}{v_{sig,i}} \frac{a}{a^{3\frac{\gamma-1}{2}}}. \quad (3.92)$$

The last factor contains the terms relating to the expansion rate of the Universe.

In EAGLE, we use an additional limiter to avoid particles getting an arbitrarily large smoothing length due to inaccurate prediction in time. The prediction of the evolution of the smoothing length is done on the basis of the divergence of the velocity field of the gas and can be quite noisy. To avoid spurious values, we forbid particles to be inactive for a time over which their smoothing length would grow by more than a given factor. This constraint can be expressed in the form of a maximal timestep:

$$\Delta t_i = 3 \ln(\alpha) \cdot (\nabla \cdot \mathbf{v})_i \cdot \frac{a}{a^{3\frac{\gamma-1}{2}}}. \quad (3.93)$$

α is the maximal change in h allowed and is called MaxSmoothingLengthChange in the code. A value of 1.26 will imply that a particle cannot change its volume by more than a factor of $2 \approx 1.26^3$ over the course of one single time step.

The smallest of those two time steps is then used as the time step of this gas particle.

3.2 Anarchy

3.2.1 Hydrodynamics

Main equations

Given the discrete quantity x_i , its smoothed, SPH value (namely its density) at position \mathbf{r}_i is

$$y_i = \sum_{j=1}^N x_j W_{ij} \quad (3.94)$$

where the sum is over N neighbours, and

$$W_{ij} \equiv W(|\mathbf{r}_i - \mathbf{r}_j|, h_i), \quad (3.95)$$

is the SPH kernel. Therefore, one can define the volume associated to x_i as

$$V_i = \frac{x_i}{y_i} \quad (3.96)$$

For example the SPH density is

$$\rho_{\text{SPH}} = \sum_j m_j W_{ij}. \quad (3.97)$$

The generalised, SPH equation of motion is (Hopkins, 2012)

$$m_i \frac{d\mathbf{v}_i}{dt} = \sum_{j=1}^N P_j \left(\nabla_i V_j + \psi_j \nabla_i \tilde{V}_j \right) \quad (3.98)$$

where ψ_j is the correction term for variable smoothing length

$$\psi_j = \frac{h_j}{n_D \tilde{V}_j} \frac{\partial V_j}{\partial h_j} \left[1 - \frac{h_j}{n_D \tilde{V}_j} \frac{\partial \tilde{V}_j}{\partial h_j} \right]^{-1} \quad (3.99)$$

and n_D is the number of spatial dimensions; $\tilde{V}_j = \tilde{x}_j/\tilde{y}_j$ is the particle volume, which defines the number of neighbours, thus the smoothing length; $V_j = x_j/y_j$

Substituting ψ_j , \tilde{V}_j and V_j with their definitions, equation ?? can be written as

$$m_i \frac{d\mathbf{v}_i}{dt} = - \sum_{j=1}^N x_i x_j \left[\frac{P_i}{y_i^2} f_{ij} \nabla_i W_{ij} + \frac{P_j}{y_j^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.100)$$

where

$$f_{ij} = 1 - \frac{\tilde{x}_j}{x_j} \left(\frac{h_i}{n_D \tilde{y}_i} \frac{\partial y_i}{\partial h_i} \right) \left[1 + \frac{h_i}{n_D \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial h_i} \right]^{-1} \quad (3.101)$$

Assuming $\tilde{x}_i = m_i$ (the particle mass), we obtain $\tilde{y}_i = \rho_i$ (the particle density), and the particle volume is then

$$\tilde{V}_i = m_i / \rho_i \quad (3.102)$$

We define the variable

$$A_i^{1/\gamma} \equiv \frac{P^{1/\gamma}}{\rho}, \quad (3.103)$$

where A_i is the particle entropy (or entropic function), so $P = A\rho^\gamma$. Assuming $x_i = m_i A_i^{1/\gamma}$ and $y_i = \sum_{j=1}^N m_j A_j^{1/\gamma}$, we can define the volume

$$V_i = \frac{m_i A_i^{1/\gamma}}{\sum_{j=1}^N m_j A_j^{1/\gamma} W_{ij}} = \frac{m_i}{A_i^{-1/\gamma} \sum_{j=1}^N m_j A_j^{1/\gamma} W_{ij}} \quad (3.104)$$

As one can immediately see,

$$\rho_A(i) \equiv \bar{\rho}_i = \frac{\sum_{j=1}^N m_j A_j^{1/\gamma} W_{ij}}{A_i^{1/\gamma}} = \frac{y_i}{A_i^{1/\gamma}} = m_i \frac{y_i}{m_i A_i^{1/\gamma}} = m_i \frac{y_i}{x_i} \quad (3.105)$$

has the dimensions of a density. Moreover, y_i defines the smoothed pressure through the relation

$$\bar{P} = y_i^\gamma \quad (3.106)$$

Rewriting equation ?? as

$$m_i \frac{d\mathbf{v}_i}{dt} = - \sum_{j=1}^N x_i x_j \left[\frac{P_i}{y_i^2} f_{ij} \nabla_i W_{ij} + \frac{P_j}{y_j^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.107)$$

$$= - \sum_{j=1}^N x_i x_j \left[\frac{m_i^2 x_i^2}{m_i^2 x_i^2 y_i^2} \frac{P_i}{y_i^2} f_{ij} \nabla_i W_{ij} + \frac{m_j^2 x_j^2}{m_j^2 x_j^2 y_j^2} \frac{P_j}{y_j^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.108)$$

$$= - \sum_{j=1}^N m_i m_j \frac{x_i}{m_i} \frac{x_j}{m_j} \left[\frac{m_i^2}{x_i^2} \frac{P_i}{(m_i y_i / x_i)^2} f_{ij} \nabla_i W_{ij} + \frac{m_j^2}{x_j^2} \frac{P_j}{(m_j y_j / x_j)^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.109)$$

$$= - \sum_{j=1}^N m_i m_j \left[\frac{m_i}{x_i} \frac{x_j}{m_j} \frac{P_i}{(m_i y_i / x_i)^2} f_{ij} \nabla_i W_{ij} + \frac{x_i}{m_i} \frac{m_j}{x_j} \frac{P_j}{(m_j y_j / x_j)^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.110)$$

With a simple substitution we obtain

$$\frac{d\mathbf{v}_i}{dt} = - \sum_{j=1}^N m_j \left[\frac{A_j^{1/\gamma} \bar{P}_i}{A_i^{1/\gamma} \bar{\rho}_i^2} f_{ij} \nabla_i W_{ij} + \frac{A_i^{1/\gamma} \bar{P}_j}{A_j^{1/\gamma} \bar{\rho}_j^2} f_{ji} \nabla_j W_{ij} \right] \quad (3.111)$$

The terms f_{ij} and f_{ji} are given by equation ??:

$$f_{ij} = 1 - \frac{\tilde{x}_j}{x_j} \left(\frac{h_i}{n_D \tilde{y}_i} \frac{\partial y_i}{\partial h_i} \right) \left[1 + \frac{h_i}{n_D \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial h_i} \right]^{-1} \quad (3.112)$$

$$= 1 - \frac{1}{A_j^{1/\gamma}} \left(\frac{h_i}{n_D \rho_i} \frac{\partial \bar{P}_i^{1/\gamma}}{\partial h_i} \right) \left[1 + \frac{h_i}{n_D \rho_i} \frac{\partial \rho_i}{\partial h_i} \right]^{-1} \quad (3.113)$$

$$= 1 - \frac{1}{A_j^{1/\gamma}} \zeta_{P,i} \zeta_{D,i} = 1 - \frac{A_i^{1/\gamma}}{A_j^{1/\gamma}} \tilde{f}_i \quad (3.114)$$

where

$$\tilde{f}_i = \frac{1}{A_i^{1/\gamma}} \zeta_{P,i} \zeta_{D,i} \quad (3.115)$$

and

$$\zeta_{P,i} = \frac{h_i}{n_D \rho_i} \frac{\partial \bar{P}_i^{1/\gamma}}{\partial h_i} \quad \text{and} \quad \zeta_{D,i} = \left[1 + \frac{h_i}{n_D \rho_i} \frac{\partial \rho_i}{\partial h_i} \right]^{-1} \quad (3.116)$$

Substituting in equation ??,

$$\frac{d\mathbf{v}_i}{dt} = - \sum_{j=1}^N m_j \left[\left(\frac{A_j^{1/\gamma}}{A_i^{1/\gamma}} - \tilde{f}_i \right) \frac{\bar{P}_i}{\bar{\rho}_i^2} \nabla_i W_{ij} + \left(\frac{A_i^{1/\gamma}}{A_j^{1/\gamma}} - \tilde{f}_j \right) \frac{\bar{P}_j}{\bar{\rho}_j^2} \nabla_j W_{ij} \right] \quad (3.117)$$

which is the equation in the code.

Alternatively, one could define the volume by using the particle number density

$$n_i = \sum_{j=1}^N W_{ij} \quad (3.118)$$

This is equivalent to using $\tilde{x} = 1$ and $\tilde{y} = n_i = \sum_{j=1}^N W_{ij}$, and the particle volume is then

$$\tilde{V} = \frac{1}{\sum_{j=1}^N W_{ij}} = \frac{1}{n_i} \quad (3.119)$$

Equation ?? remains unchanged, while the term f_{ij} becomes

$$f_{ij} = 1 - \frac{\tilde{x}_j}{x_j} \left(\frac{h_i}{n_D \tilde{y}_i} \frac{\partial y_i}{\partial h_i} \right) \left[1 + \frac{h_i}{n_D \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial h_i} \right]^{-1} \quad (3.120)$$

$$= 1 - \frac{1}{m_j A_j^{1/\gamma}} \left(\frac{h_i}{n_D n_i} \frac{\partial \bar{P}_i^{1/\gamma}}{\partial h_i} \right) \left[1 + \frac{h_i}{n_D n_i} \frac{\partial n_i}{\partial h_i} \right]^{-1} \quad (3.121)$$

$$= 1 - \frac{1}{m_j A_j^{1/\gamma}} \zeta'_{P,i} \zeta'_{D,i} = 1 - \frac{A_i^{1/\gamma}}{A_j^{1/\gamma}} \tilde{f}_{ij} \quad (3.122)$$

where

$$\tilde{f}_{ij} = \frac{1}{m_j A_i^{1/\gamma}} \zeta'_{P,i} \zeta'_{D,i} \quad (3.123)$$

It is trivial to show that, for equal mass particles,

$$\zeta_{P,i} = \zeta'_{P,i}/m_j \quad \zeta_{D,i} = \zeta'_{D,i} \quad (3.124)$$

Substituting in equation ??,

$$\frac{d\mathbf{v}_i}{dt} = - \sum_{j=1}^N m_j \left[\left(\frac{A_j^{1/\gamma}}{A_i^{1/\gamma}} - \tilde{f}_{ij} \right) \frac{\bar{P}_i}{\bar{\rho}_i^2} \nabla_i W_{ij} + \left(\frac{A_i^{1/\gamma}}{A_j^{1/\gamma}} - \tilde{f}_{ij} \right) \frac{\bar{P}_j}{\bar{\rho}_j^2} \nabla_j W_{ij} \right] \quad (3.125)$$

Implementation

Implementation details refer to svn version **24524**.

EntropyVarPred

`predict.c`

The variable `SphP[i].EntropyPred` is used as entropy variable. It is calculated from the predicted entropy in `predict.c` as

$$\text{SphP}[i].\text{EntropyVarPred} \equiv A_i^{1/\gamma} \text{ defined in Eq. ??, and calculated in } \text{predict.c} \text{ l=198.} \quad (3.126)$$

`density.c`

The ‘weighted density’ $\bar{\rho}_i$ from Eq. (??) is calculated as

$$\text{SphP}[i].\text{cky.WeightedDensity} \equiv \bar{\rho}_i \quad (3.127)$$

$$= \sum_j m_{j_wk} * \text{SphP}[j].\text{EntropyVarPred}$$

$$(l=2220) \quad (3.128)$$

$$/ \text{SphP}[i].\text{EntropyVarPred} \text{ l=1082, } (3.129)$$

where $m_{j_wk} = P[j].\text{Mass} \times W(r_{ij}/h_i)$ ($l=2205$), with the kernel evaluated in `kernel.h`.

Cooling and equation of state

The internal energy u_i is derived from the particle’s entropy $A_i(t + \Delta t) = A_i(t) + \dot{A}_i \Delta t$ and its anarchy density ρ_A as

$$u(t) = \frac{A_i(t + \Delta t) \rho_A^{\gamma-1}}{\gamma - 1}, \quad (3.130)$$

`eagle_cooling.c` l=139. Cooling integrates

$$\rho_{\text{SPH}} \frac{du}{dt} = -\Lambda_{\text{net}}(u) \rho_{\text{SPH}}^2, \quad (3.131)$$

`eagle_cooling.c` l=161, which leads to a new thermal energy u_{new} . This is converted back to a new entropy using Eq. (??).

Note that we cool at ‘constant’ ρ_A , whereas without pressure-entropy we cool at constant ρ_{SPH} , but that nevertheless ρ_A would be different if we re-computed the density.

Above a given density threshold, gas is moved onto an imposed pressure-density relation to avoid artificial fragmentation. The shape of the imposed eos, and of the minimum entropy, is

$$p = p_{\text{eos}} (\rho / \rho_{\text{eos}})^{\gamma_{\text{eos}}} \quad (3.132)$$

$$p_{\text{eos}} = (\gamma - 1) \rho_{\text{eos}} (k_{\text{B}} T_{\text{eos}} / ((\gamma - 1) \mu m_{\text{H}})) \quad (3.133)$$

$$S = \frac{p}{\rho^\gamma} = \frac{p_0}{\rho_0^\gamma} \left(\frac{\rho_0}{\rho} \right)^{\gamma - \gamma_{\text{eos}}} . \quad (3.134)$$

The input parameters that describe this are

1. the minimum physical density, $\rho_{\text{min}} = \text{EOS_Jeans_MinPhysDens_HpCM3}$
2. the minimum physical over density, $\Delta_{\text{min}} = \text{EOS_Jeans_MinOverDens}$
3. the temperature at the start of the eos, $T_{\text{min}} = \text{EOS_Jeans_TempNorm_K}$
4. the slope of the EOS, $\gamma_{\text{eos}} = \text{EOS_Jeans_GammaEffective}$

There is a similar set of parameters to impose a minimum temperature. These get written in a form such that

$$p = p_{\text{eos}} \left(\frac{\rho}{\rho_{\text{eos}}} \right)^{\gamma_{\text{eos}}} , \quad (3.135)$$

The entropy is not allowed to be lower than

$$A_i \geq p_{\text{eos}} \left(\frac{\rho_{\text{SPH}}}{\rho_{\text{eos}}} \right)^{\gamma_{\text{eos}}} \frac{1}{\rho_{\text{SPH}}^\gamma} . \quad (3.136)$$

Having obtained the new entropy, we update \dot{A}_i .

Arguably we should use ρ_{A} here, since it is this density that determines pressure forces and hence artificial fragmentation.

Star formation

The star formation rate is given by Eq. (??) and is zero if the particle is not dense enough or too hot. The limits are calculate in `eagle_sfr.c` at l=468 and l=477 for density, and l=480 for entropy:

$$\begin{aligned} \rho_{\text{SPH}} &\geq \rho_{\text{threshold}} \\ A &\leq A_{\text{max}} . \end{aligned} \quad (3.137)$$

The entropy floor is also calculated using ρ_{SPH} . Of the particle is deemed to be star forming, its star formation rate is $\propto (\frac{2p}{G})^{(n_{\text{KS}}-1)/2}$, where the pressure $p = A(t) \rho_A^\gamma$ is computed in `predict.c` l=325.

Stellar feedback

Having calculated the change in specific energy of a particle due to feedback, Δu_{cgs} , the particle's properties are updated in `anarchy_energy_injection.c`. The pre-feedback u is calculated from

$$u_{\text{old}} = \frac{A(t + \Delta t) \rho_A^{\gamma-1}}{\gamma - 1}, \quad (3.138)$$

with the new thermal energy

$$u_{\text{new}} \equiv u_{\text{old}} + \Delta u. \quad (3.139)$$

Note that `anarchy_energy_injection.c` uses programme units, not cgs units. Note that u_{old} uses the predicted entropy and the anarchy density. We then iterate updating the new entropy and density according to

$$\begin{aligned} A_{n+1} &= \frac{(\gamma - 1)u_{\text{new}}}{\rho_{A,n}^{\gamma-1}} \\ \rho_{A,n+1} &= \frac{\rho_n A_n^{1/\gamma} - mW(0)A_n^{1/\gamma} + mW(0)A_{n+1}^{1/\gamma}}{A_{n+1}^{1/\gamma}}, \end{aligned} \quad (3.140)$$

until the relative difference between ρ_{n+1} and ρ_n , and the same for A , is less than 10^{-6} , or there are more than 10 iterations. Once this is converged, we update A , ρ_A , and the pressure, $p \propto \rho_A^\gamma$. (Note there is no entropy floor check here, since presumably entropy increases.) Note that this is not correct, in the sense that the change in thermal energy (in the whole box) is not $m\Delta u$.

Black hole accretion and feedback

We start by calculating an ‘entropy’ at the location of a black hole by calcu-

lating

$$\begin{aligned}
\rho_{\text{BH,SPH}} &\equiv \text{BH.b1.dBH_Density} = \sum_j m_j W_j \quad (\text{l}=2207, 479) \\
A_{\text{BH}} &\equiv \text{BH.b2.dBH_Entropy} = \sum_j m_j W_j A(t)_j \quad (\text{l}=2539, 480) \\
&\quad / \rho_{\text{BH,SPH}} \quad (\text{l}=1302) \\
\rho_A &\equiv \text{BH.b11.dBH_Weighted_Density} = \sum_j m_j W_j A_j^{1/\gamma} \quad (\text{l}=2220, 483) \\
&\quad / A_{\text{BH}}^{1/\gamma} \quad (\text{l}=1308). \quad (3.141)
\end{aligned}$$

The sound speed and density for use in the accretion rate are

$$\begin{aligned}
c_s &\equiv \text{soundspeed} = (\gamma A_{\text{BH}} \rho_{\text{BH,SPH}}^{\gamma-1})^{1/2} \quad (\text{l}=199) \\
\dot{m} &\propto \frac{m^2 \rho_{\text{BH,SPH}}}{(c_s^2 + v^2)^{3/2}} \quad (\text{l}=210)
\end{aligned}$$

It is the value of c_s that is reported in the log file. Energy injection uses the same routine as stellar thermal feedback.

3.2.2 Artificial viscosity

3.2.3 Entropy diffusion

3.2.4 Energy injection

3.2.5 Black hole growth and feedback

Implementation details refer to SVN version 24344

Gas properties at BH position

In the following, the subscript BH identifies local gas properties at the black hole position.

In Anarchy, the conversion from entropy to internal energy needs the (entropy) weighted density, $\bar{\rho}$, defined in equation (??). Therefore, to compute the local sound speed, $c_{\text{s,BH}}$, the black hole structure must hold the weighted density, $\bar{\rho}_{\text{BH}}$, and the local (average) entropy, A_{BH} .

The density is computed in `density.c` [line 2197] as done for gas particles, and stored in `BPP(i).b11.BH_Weighted_Density` [lines 483, 1040, 1308].

The *mass weighted* entropy at BH position is also computed in `density.c` [line 2197], and stored in `BPP(i).b2.BH_Entropy` [lines 480, 1036, 1302].

In `eagle_blackhole.c`, the sound speed is computed as follows. Depending on whether PE-SPH is used or not, the variable `weighted_density` holds either `BPP(i).b11.BH_Weighted_Density` or `BPP(i).b1.BH_Density` [lines 183-188]. Note that the density is physical, and converted with the proper a^{-3} factor. Therefore, the black hole local sound speed is simply [line 199]

$$\text{soundspeed} = \sqrt{\text{GAMMA} * \text{BPP}(n).\text{b2.BH_Entropy} * \text{pow}(\text{weighted_rho}, \text{GAMMA_MINUS1})}, \quad (3.142)$$

(see also [lines 357-365, 738-746]). This are all the changes needed for black holes growth.

Black hole feedback

The only change in `eagle_bhfeedback.c` regards the injection into gas particles of the black hole feedback energy. This is done by using the function `anarchy_energy_injection(j, delta_u)` [line 619], where `j` is the gas particle index, and `delta_u` is the injected energy in physical, code units. The description of the injection method is below.

Issues

When compiled with `EAGLE_DETACH_BH_DENSITY` the code does not compute the black hole weighted density in `eagle_blackhole_density.c`. The calculation is indeed missing in `eagle_blackhole.c`. However, this function is redundant, as gas properties around the black holes are computed in `density.c` regardless of the above compilation flag.

3.2.6 Feedback energy injection

Internal energy changes due to feedback are implemented by using the function `anarchy_energy_injection(i, delta_u)` in `anarchy/anarchy_energy_injection.c`.

The function takes the particle index i and the amount of internal energy change `delta_u` (Δu) as arguments.

Energy injection proceeds iteratively by computing the new value of the entropy, and adjusting the weighted density (which is a function of entropy) to it. We know the current value of the particle internal energy, $u_i = \frac{1}{\gamma-1} A_i \bar{\rho}_i^{\gamma-1}$, and we know its value, $u'_i = u_i + \Delta u$, after the feedback event. We also know the initial weighted density, $\bar{\rho}_{i,0}$, and entropy, $A_{i,0}$. The iterative method follows:

1. a guess of the new entropy is computed as $A_{i,1} = (\gamma - 1) u'_i \bar{\rho}_{i,0}$;
2. the weighted density is corrected according to the change to the particle weight in the density estimate, $m_i A_i$:

$$\bar{\rho}_{i,1} = \frac{1}{A_{i,1}^{1/\gamma}} \left[\bar{\rho}_{i,0} A_{i,0}^{1/\gamma} - m_i (A_{i,0}^{1/\gamma} - A_{i,1}^{1/\gamma}) W_{ii}(0, h_i) \right]; \quad (3.143)$$

3. convergence is checked for both density and entropy, requiring that the variation from the previous iteration is smaller than **TOLERANCE**.
4. if the above criterion is not verified and the number of iteration is smaller than **MAX_ITER**, set $\bar{\rho}_{i,0} = \bar{\rho}_{i,1}$ and $A_{i,0} = A_{i,1}$ and restart from (1).

The values of **TOLERANCE** and **MAX_ITER** are 10^{-6} and 10, respectively.

Issues

The problem with the above method is that it does not take into account variations of internal energy of nearby particles. This may results in wrong estimates of the weighted densities. The best solution is to use the pressure-energy formulation of SPH.

Let's assume that $\Delta u \gg u_j$, and only particle i receives energy. The self-contribution term in the computation of the weighted density will dominate the sum, and we can then approximate the particle weighted density to

$$\bar{\rho}_i = \frac{1}{A_i^{1/\gamma}} \sum_{j=1}^N m_j A_j^{1/\gamma} W_{ij} \simeq m_i W_{ii}. \quad (3.144)$$

This is what happens applying the above iterative method. Moreover, when several neighbours receive energy, the change of their entropy is not accounted.

On the other hand, if n neighbours received the same large amount of energy, to some order, the weighted density of particle i should become

$$\bar{\rho}'_i = \frac{1}{A_i^{1/\gamma}} \sum_{j=1}^N m_j A_j^{1/\gamma} W_{ij} \simeq \frac{1}{A_i^{1/\gamma}} \sum_{k=1}^n m_k A_k^{1/\gamma} W_{ik} \sim n m_i W_{ii}. \quad (3.145)$$

Let's now suppose that, at a give time, the entropy of particle i is modified by using the iterative method. The particle has now density $\bar{\rho}_i$, as in equation (??). The particle internal energy, just after energy injection, is then

$$u_i = \frac{1}{\gamma - 1} A_i \bar{\rho}_i^{\gamma-1}. \quad (3.146)$$

The internal energy is conserved, but only with the particular combination of density and entropy in which neighbours are assumed to keep their entropy unchanged during the same time-step.

Let's suppose the above assumption does not hold, and several to many neighbours of particle i changed their entropy during the same time-step. If particle i becomes active in the next time-step, its density is updated in **density.c**. Its new density is $\bar{\rho}'_i$, as in equation (??). This can be much larger than $\bar{\rho}_i$, as several to many neighbours of particle i have just been affected by feedback (typical case in black hole feedback). Therefore, the internal energy of particle i is

$$u'_i = \frac{1}{\gamma - 1} A_i \bar{\rho}'_i^{\gamma-1} \gg u_i. \quad (3.147)$$

This may explain why black hole feedback is so efficient in pressure-entropy runs.

3.2.7 Effective equation of state

3.2.8 Star formation rate

3.2.9 Cooling

3.3 Cooling

In EAGLE, cooling and photo-heating rates are interpolated from tables. At higher densities, particles are required to have a minimum entropy, which depends on density. One is to impose a given $p - \rho$ relation, $p = p_0 (\rho/\rho_0)^{\gamma_{\text{eos}}}$, or equation of state, mimicking ISM gas, and at even higher density there is a limit to make sure the Jeans mass, M_J , keeps being resolved with the same number of particles. Since $M_J \propto T^{3/2}/\rho^{1/2}$, this corresponds to $\gamma_{\text{eos}} = 4/3$. The imposed eos has then 4 parameters specified in the parameter file:

- ρ_{eos} , the physical density above which the eos is imposed
- Δ_{eos} , a minimum over density above which the eos is imposed
- T_{eos} , the temperature at the eos threshold
- γ_{eos} , the slope of the imposed $p - \rho$, relation

The shape of the eos, and of the minimum entropy, is thus

$$p = p_{\text{eos}} (\rho/\rho_{\text{eos}})^{\gamma_{\text{eos}}} \quad (3.148)$$

$$p_{\text{eos}} = (\gamma - 1) \rho_{\text{eos}} (k_B T_{\text{eos}} / ((\gamma - 1) \mu m_H)) \quad (3.149)$$

$$S = \frac{p}{\rho^\gamma} = \frac{p_0}{\rho_0^\gamma} \left(\frac{\rho_0}{\rho} \right)^{\gamma - \gamma_{\text{eos}}} . \quad (3.150)$$

The eos is imposed at relatively high density, where the temperature is so high the gas is mostly (collisionionally) ionised. Hence the code assumes the mean molecular weight to be $\mu = 0.59$, which is hard-coded in terms of a defined parameter MEANMOLIONIZED in *allvars.h*.

3.4 Star formation

See Schaye & Dalla Vecchia 08. Assume a Kennicutt-Schmidt law of the form

$$\Sigma_\star = \begin{cases} 0, & \text{for } \Sigma_g < \Sigma_c \\ A_{\text{KS}} \left(\frac{\Sigma_g}{1 M_\odot \text{ pc}^{-2}} \right)^{n_{\text{KS}}}, & \text{for } \Sigma_g \geq \Sigma_c \end{cases} \quad (3.151)$$

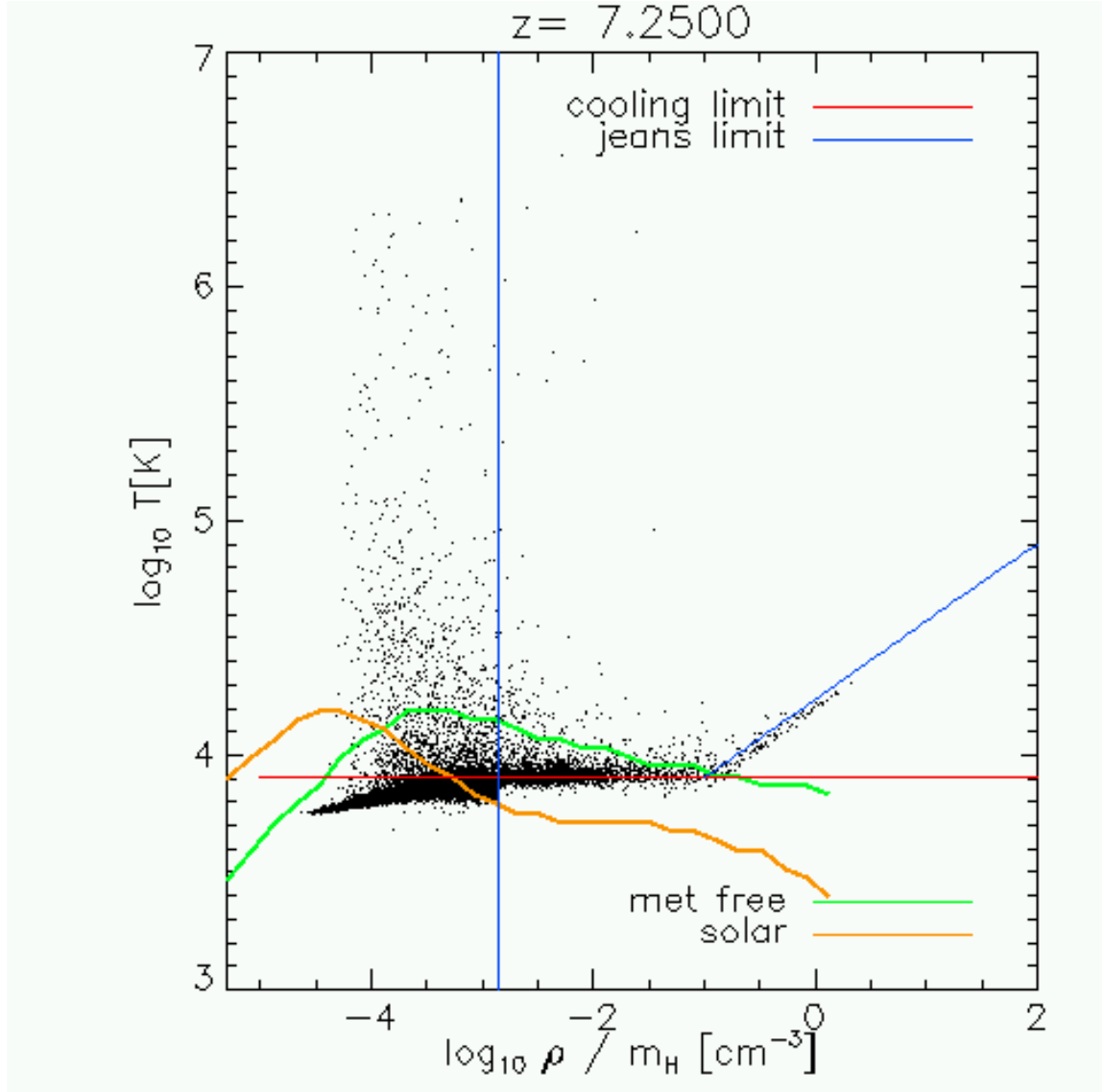


Figure 3.1: Temperature-density relation as $z = 7.25$, for a *L003N063* run with assumed abundance of $0.1 Z_{\odot}$ gas. Green and yellow curves are the equilibrium temperatures for metal free and solar abundances, respectively. Imposed ‘cooling’ and ‘Jeans’ EOS are shown by blue and red lines, respectively. Vertical lines denote the over density threshold (which is the same for both), and an imposed eos runs from ρ_{eos} to high densities. In this case you see the over-density limit, which does not allow the gas to be below the horizontal blue line, once it passes the vertical blue line, which sets in at $\rho \approx 10^{-3} \text{ cm}^{-3}$. At higher density, $\rho = 10^{-1} \text{ cm}^{-3}$, the imposed ‘Jeans’ eos takes over. Note that the particles do not follow the imposed lines exactly, because the particle’s mean molecular weight is not exactly equal to `MEANMOLIONIZED`.

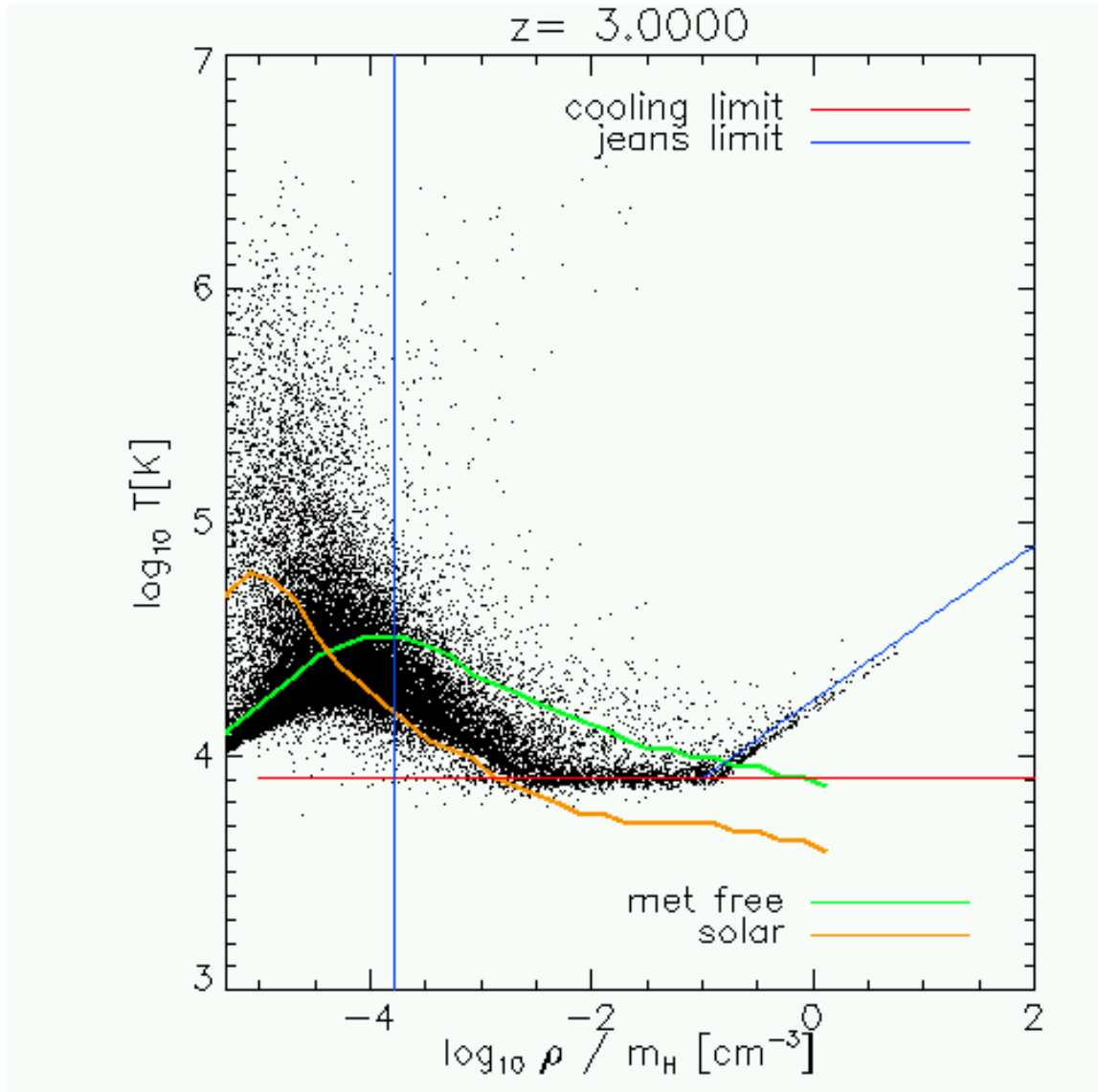


Figure 3.2: Same as Fig.??, but at $z = 3.0$. Note how gas at $\rho \sim 10^{-4} \text{ cm}^{-3}$ tracks just above the solar equilibrium curve.

where $A_{\text{KS}} = (2.5 \pm 0.7) \times 10^{-4} M_{\odot} \text{ yr}^{-1} \text{ kpc}^{-2}$ and $n_{\text{KS}} = 1.4 \pm 0.15$. In a hydrostatic disc galaxy this requires a specific star formation rate of

$$\frac{\dot{m}_{\star}}{m_g} \equiv \frac{1}{t_{\star}} = A_{\text{KS}} (1 M_{\odot} \text{ pc}^{-2})^{-n_{\text{KS}}} \left(\frac{\gamma p}{G}\right)^{(n_{\text{KS}}-1)/2}, \quad (3.152)$$

where numerically

$$t_{\star} = 1.67 \times 10^9 \text{ yr} \left(\frac{p/k_B}{10^3 \text{ cm}^{-3} \text{ K}}\right)^{-0.2}, \quad (3.153)$$

for $n_{\text{KS}} = 1.4$ and $A_{\text{KS}} = 2.5 \times 10^{-4} M_{\odot} \text{ yr}^{-1} \text{ kpc}^{-2}$. Above a threshold density $\rho_{\star, \text{KS}}$, the index of the power law steepens to n'_{KS} .

In EAGLE gas particles are only star forming, within limits on physical density, over density and entropy. If the particle falls within these limits, its ‘eos’ flag = 1 and the particle is eligible for star formation. In practise

- $\rho > \rho_{\star}$: the physical density needs to be sufficiently high
- $\Delta > \Delta_{\star}$: the over density needs to be sufficiently high
- $S < S_{\text{eos}} 10^{\Delta T_{\text{dex}}}$: the gas needs to be sufficiently cold.

The physical threshold ρ_{\star} depends on metallicity as

$$\rho_{\star} > \min \left\{ \rho_{0,\star}, \rho_{m,\star} \left(\frac{Z}{0.1 \times 0.02}\right)^{\alpha_z} \right\} \quad (3.154)$$

Eq. (??) is implemented in routine *eagle_get_eos_flag*: Z is the particle’s metallicity. For $Z = 0$ we use a fixed threshold $\rho_{0,\star}$, for $Z > 0$, the threshold increases $\propto Z^{\alpha_z}$. At $z = 0.1 \times 0.02$ (or ‘ten per cent solar’), the threshold is $\rho_{m,\star}$.

3.5 Stellar feedback

3.5.1 Virial temperature calculation

T_{vir} from dark matter velocity dispersion

Routine *eagle_stellar_feedback_set_tvir_from_dm_velocity_dispersion* calculated the DM velocity dispersion of active stars, within a smoothing length. The

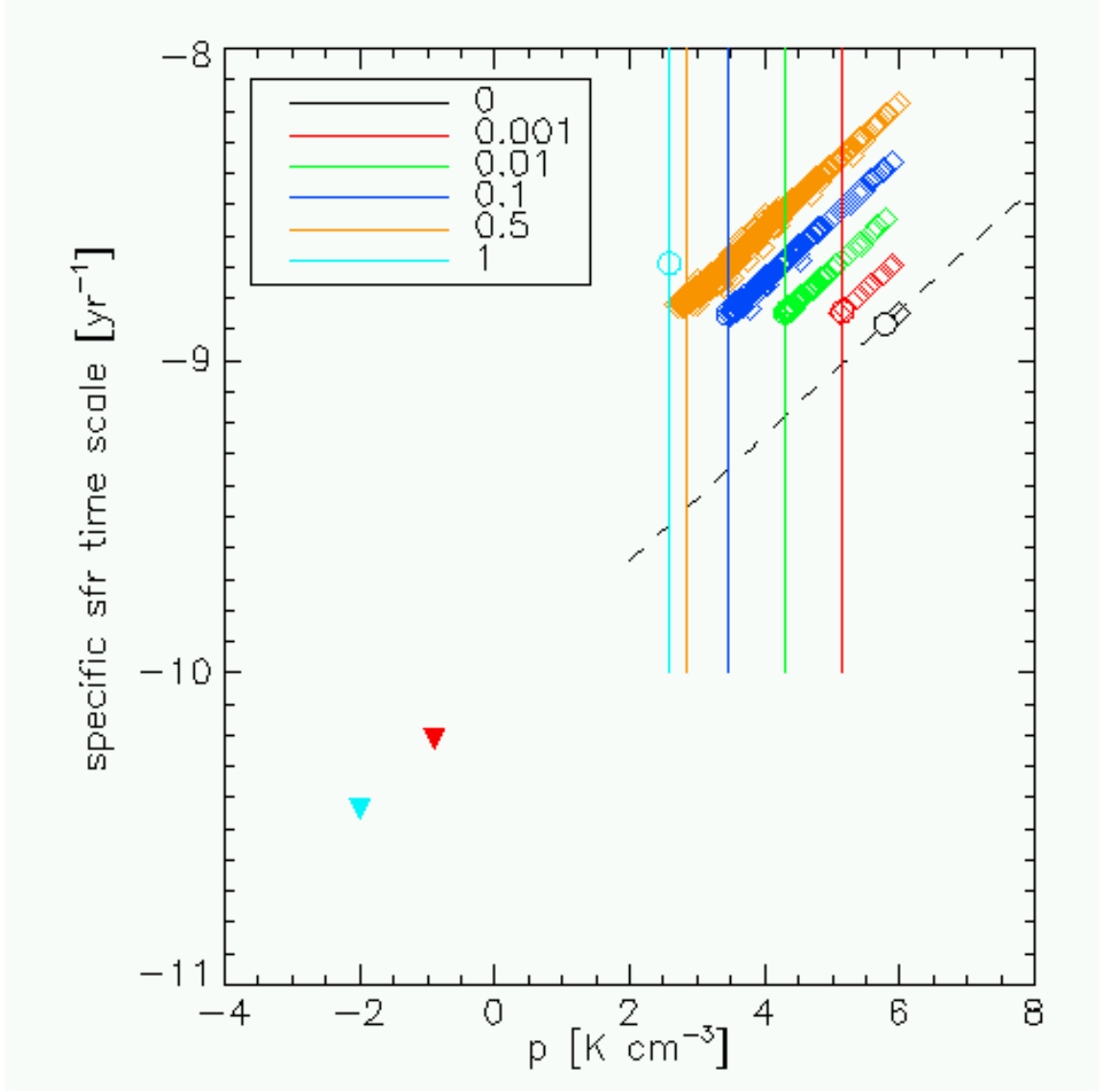


Figure 3.3: Specific star formation rate, $\dot{\rho}_*/\rho$, as function of pressure of the gas. Dashed line is the imposed KS-law, with thresholds for star formation imposed: cyan triangle: over density threshold, red triangle: imposed eos threshold, circles: thresholds for abundances of $[1, 0.5, 10^{-1}, 10^{-2}, 10^{-3}, 0] \times Z_\odot$, respectively cyan, orange, blue, green, red, and black. Squares are results from the simulation, with the same colours. Different metallicity points are offset vertically for clarity. The sfr computed in the simulation follows the imposed KS-law and metallicity thresholds.

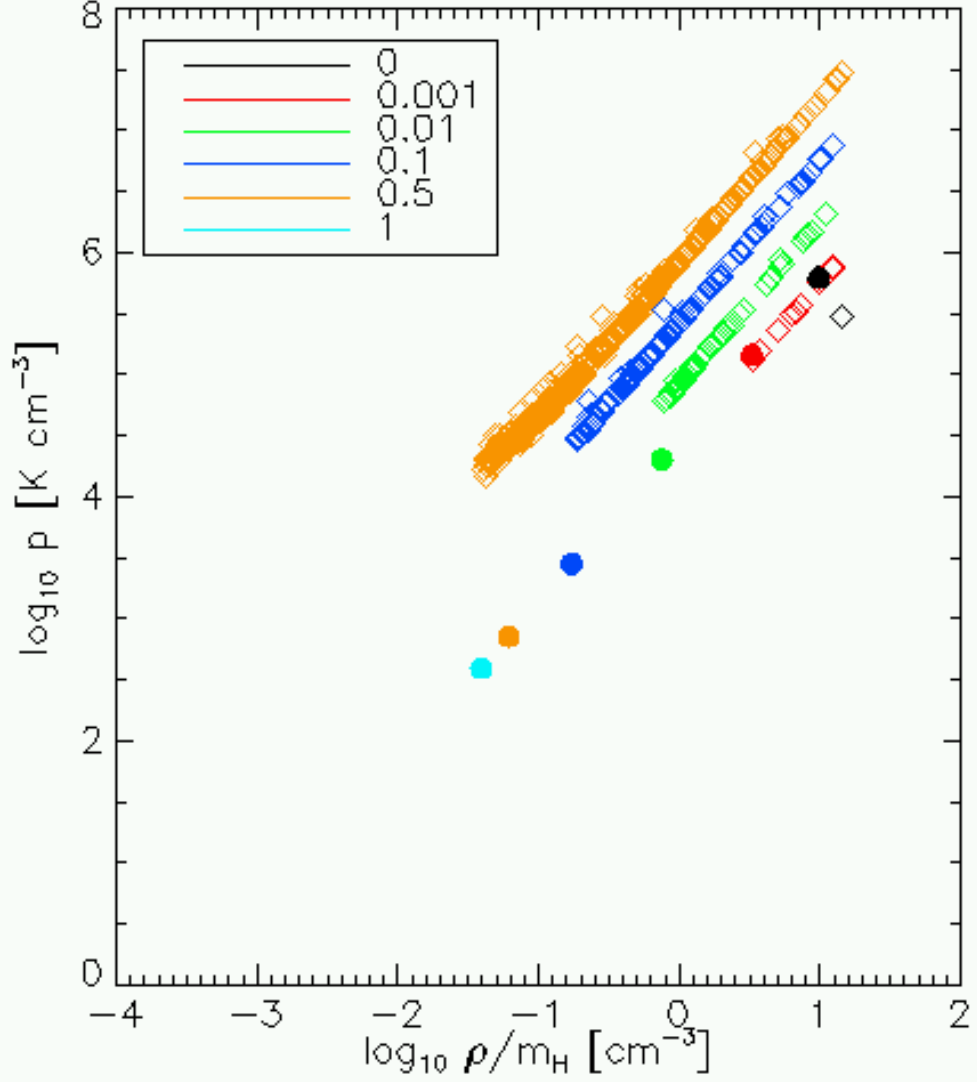


Figure 3.4: ρ – p relation along the imposed equation of state. Star formation density thresholds are indicated in colour (circles) for various metallicities in units of Z_{\odot} . Squares are results from the simulation, with the same colours. Different metallicity points are offset vertically for clarity. Star formation follows the imposed EOS, as well as the imposed metallicity dependent density thresholds.

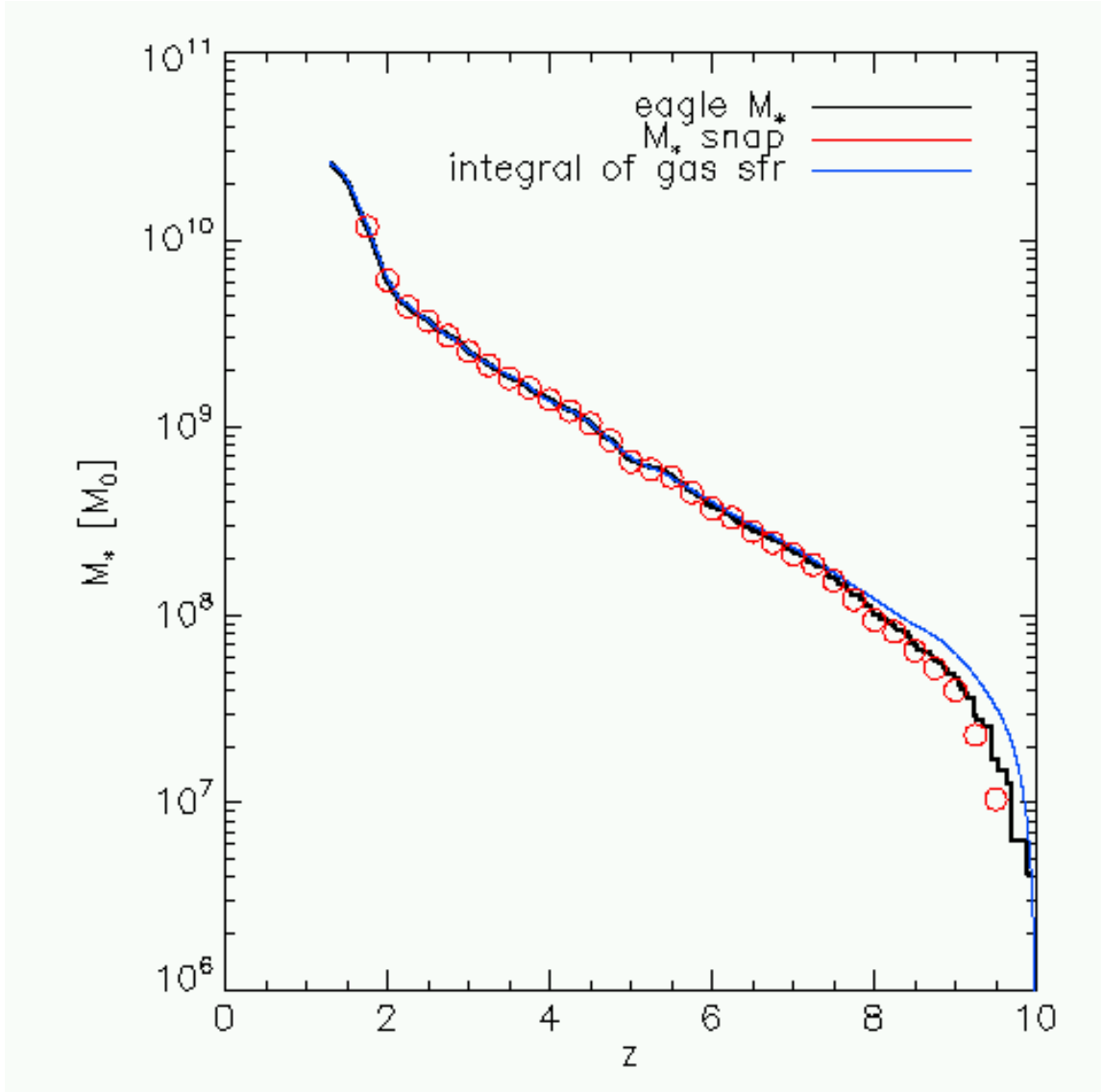


Figure 3.5: Total (initial) stellar mass as function of redshift, as computed in eagle_sfr (black), from the snapshots (red), and obtained by integration the computed star formation rate for gas particles (blue).

smoothing length itself is not stored, and calculated only once, as the radius within which the number of (kernel-weighted) neighbours is `NNB_Mult` times `All.DesNumNgb`. `NNB_Mult` is defined in the file itself (`eagle_stellar_feedback_set_tvir`) and `All.DesNumNgb` is an input parameter. It may make sense to decrease the noisy estimate of the velocity dispersion by increasing the number of weighted numbers above the default 48 by taking `NNB_Mult=4`, for example. The routine calculates the standard deviation of the DM velocities (in code units), then converts this to a **physical** value, σ_v , and uses it to assign a virial temperature as (see `eagle_compute_tvir_from_veldisp`)

$$T_{\text{vir}} = \frac{\mu_{\text{ion}} m_{\text{proton}}}{3 k_{\text{B}}} \sigma_v^2, \quad (3.155)$$

where $\mu_{\text{ion}} = 4/(8 - 5 * 0.248)$ is defined as the mean molecular weight for ionised primordial gas with a He abundance of 0.248 by mass (defined in `allvars.h`), $m_{\text{proton}} = 1.6726 \times 10^{-24}$ is the proton mass in g (defined in `allvars.h`), $k_{\text{B}} = 1.38066 \times 10^{-16}$ is Boltzmann's constant in erg K⁻¹ (defined in `allvars.h`).

T_{vir} from virial mass, M_{200} and R_{200} .

The virial mass, M_{200} , is defined such that the mean density within the corresponding virial radius, R_{200} is 200 times the critical, density, ρ_c , hence

$$\frac{M_{200}}{(4\pi/3) R_{200}^3} = 200 \rho_c = 200 \frac{3H^2}{8\pi G} \quad (3.156)$$

$$R_{200} = \left(\frac{G M_{200}}{100 H^2} \right)^{1/3}, \quad (3.157)$$

This leads to a definition of the virial temperature, by using the virial theorem to demand that

$$2 M_{200} \frac{3 k_{\text{B}} T}{2 \mu m_{\text{proton}}} = \frac{G M_{200}^2}{R_{200}}. \quad (3.158)$$

Combining these gives

$$T_{\text{vir}} = \frac{\mu_{\text{ion}} m_{\text{proton}}}{3 k_{\text{B}}} (10 G H(z) M_{200})^{2/3}, \quad (3.159)$$

where M_{200} is the halo mass in g, $G = 6.672 \times 10^{-8}$ Newton's gravitational constant in g⁻¹ cm³ s⁻² (defined in `allvars.h`), $H(z)$ is the Hubble constant at

redshift z , is calculated as $H(z) = \text{HUBBLE All.HubbleParam } E(z)$, where $\text{HUBBLE} = 3.2407789 \times 10^{18} = 100 \text{ km Mpc}^{-1} \text{ in s}$ (defined in `allvars.h`), $E(z) = (\Omega_m (1+z)^3 + \Omega_\Lambda)^{1/2}$ is computed in `set_cosmo_factors_for_current_time`, and $h = \text{All.HubbleParam}$ is the Hubble parameter. For reference, $H(z)$ and $E(z)$ are available as `All.cf_Hz` and `All.cf_Ez`, set in that same routine.

Relation between σ_v and M_{200}

Combining the two expression for T_{vir} yields the following relation between (3D) velocity standard deviation σ_v and halo mass M_{200} :

$$\sigma_v^2 = (10 G H(z) M_{200})^{2/3} = \frac{G M_{200}}{R_{200}}, \quad (3.160)$$

which is again the virial theorem: $2 \times (1/2) M_{200} \sigma^2 = G M_{200}^2 / R_{200}$.

Virial temperature derived from the FOF mass

The code does not use M_{200} but instead uses M_{FOF} . The friends-of-friends mass and corresponding radius are such that the density at the edge of the FOF halo is b^{-3} times the mean matter density $\bar{\rho}_m$, where $b \sim 0.2$ is the linking length. The mean matter density is

$$\bar{\rho}_m = \Omega_m \rho_c = \Omega_m \frac{3 H_0^2}{8 \pi G} (1+z)^3. \quad (3.161)$$

To estimate M_{FOF} we need to assume a density profile. Assume this is isothermal, $\rho(r) = \rho_{\text{FOF}} (R_{\text{FOF}}/r)^2$. Then

$$M_{\text{FOF}} = 4 \pi \rho_{\text{FOF}} R_{\text{FOF}}^3, \quad (3.162)$$

and since $\rho_{\text{FOF}} = b^{-3} \bar{\rho}_m$ is the density at the FOF halo's edge, we find $M_{\text{FOF}} / (4 \pi / 3 R_{\text{FOF}}^3) = 3 \rho_{\text{FOF}} = 3 \bar{\rho}_m b^{-3}$.

In this case, we find

$$R_{\text{FOF}} = \left(\frac{G M_{\text{FOF}}}{(3/2 b^3) \Omega_m H_0^2 (1+z)^3} \right)^{1/3}. \quad (3.163)$$

We can now define a fudge parameter α such that

$$2 \frac{3 k_B T_{\text{vir}}}{2 \mu m_{\text{proton}}} = \alpha \frac{G M_{\text{FOF}}}{R_{\text{FOF}}}. \quad (3.164)$$

Following the derivation as above then gives

$$T_{\text{vir}} = \alpha \frac{\mu_{\text{ion}} m_{\text{proton}}}{3 k_{\text{B}}} \left(\frac{3}{(2b^3)^{1/2}} G M_{\text{FOF}} \Omega_m^{1/2} H_0 (1+z)^{3/2} \right)^{2/3}. \quad (3.165)$$

Comparing with Eq. (??) shows that the redshift dependence is the same in an EdS Universe ($H(z) = \sqrt{\Omega_m} H_0 (1+z)^{3/2}$ for the high- z approximation).

For backwards compatibility we can choose α such that the virial temperatures are the same, *i.e.* that the values of the virial temperature obtained from Eq. (??) and Eq. (??) are the same at a given redshift z .

In terms of $H(z) \equiv H_0 E(z)$, this yields a value of α of

$$\alpha = \left(\frac{10 E(z)}{(3/2b^3)^{1/2} \Omega_m^{1/2} (1+z)^{3/2}} \right)^{2/3} \quad (3.166)$$

$$= 0.89, \quad (3.167)$$

for $b = 0.2$, $\Omega_m = 0.272$, $\Omega_\Lambda = 1 - \Omega_m$ and $z = 1$.

3.5.2 Energy fraction

We want to make the fraction of SN energy that goes into feedback depend on a property of the halo. The *short-drop* family of models uses a dependence directly on halo mass, M_{FOF} only, as follows

$$\epsilon(M_{\text{FOF}}) = \left(\frac{\log M_{\text{FOF}}}{\log M_c} \right)^s \quad (3.168)$$

$$\epsilon_{\text{min}} < \epsilon < \epsilon_{\text{max}}, \quad (3.169)$$

where $M_c = 10^{11} M_\odot$, $s = -26.463$, $\epsilon_{\text{min}} = 0.1$ and $\epsilon_{\text{max}} = 1$ are input parameters, with the numerical values corresponding to short drop. Note especially the *ratio of logs*, as opposed to the log of a ratio M/M_c . This curve is shown in Fig.??, together with a ‘linear’ fit

$$\epsilon = \epsilon_0 + p \log\left(\frac{M_{\text{FOF}}}{M_0}\right) \quad (3.170)$$

$$\epsilon_0 = 0.8 \quad (3.171)$$

$$\log M_0 = \epsilon_0^{1/s} \log M_c = 11.0391 \quad (3.172)$$

$$p = (\epsilon_{\text{max}} - \epsilon_{\text{min}}) / (\log M_{\text{min}} - \log M_{\text{max}}) = -0.9. \quad (3.173)$$

Here, $\epsilon(\log M_{\min, \max}) = \epsilon_{\max, \min}$, and the numerical values correspond to the short drop model.

We want to write this fit in terms of T_{vir} instead of mass. Using Eq. (??), this leads us to define T_0 as

$$T_0 = \alpha \frac{\mu_{\text{ion}} m_{\text{proton}}}{3 k_{\text{B}}} \left(\frac{3}{(2b^3)^{1/2}} G M_0 \Omega_m^{1/2} H_0 \right)^{2/3} \quad (3.174)$$

$$= 1.1 \times 10^5 \text{K}, \quad (3.175)$$

in which case

$$\epsilon(T_{\text{vir}}) = \epsilon_0 + \frac{3p}{2} \log\left(\frac{T_{\text{vir}}}{T_0}\right). \quad (3.176)$$

Note that this relation is now redshift dependent, since $T_{\text{vir}} \propto M_{\text{FOF}}^{2/3} (1+z)$. The value of T_0 is chosen such that the relations are the same at $z = 0$. To make the relation redshift independent (*i.e.* get the result from Eq.??), replace the pervious relation by

$$\epsilon(T_{\text{vir}}, z) = \epsilon_0 + \frac{3p}{2} \log\left(\frac{T_{\text{vir}}}{T_0 (1+z)}\right). \quad (3.177)$$

It might be worth generalising the last redshift dependence to give us more freedom in tuining the feedback, by introducing a new parameter β as

$$\epsilon(T_{\text{vir}}, z) = \epsilon_0 + \frac{3p}{2} \log\left(\frac{T_{\text{vir}}}{T_0 (1+z)^\beta}\right). \quad (3.178)$$

A value of $\beta = 0$ corresponds to feedback dependent on virial temperature, $\beta = 1$ corresponds to feedback depending in FOF mass.

3.5.3 Type II SNe

3.5.4 Type I SNe

3.5.5 AGB stars

As these stars transfer mass, they also transfer energy since the velocities \mathbf{v}_\star and \mathbf{v}_{gas} of star and gas particle are not the same. We therefore want to add momentum, kinetic energy, and potentially thermal energy, of the accreted particles to the accreting particle. Let the initial mass, velocity, and specific

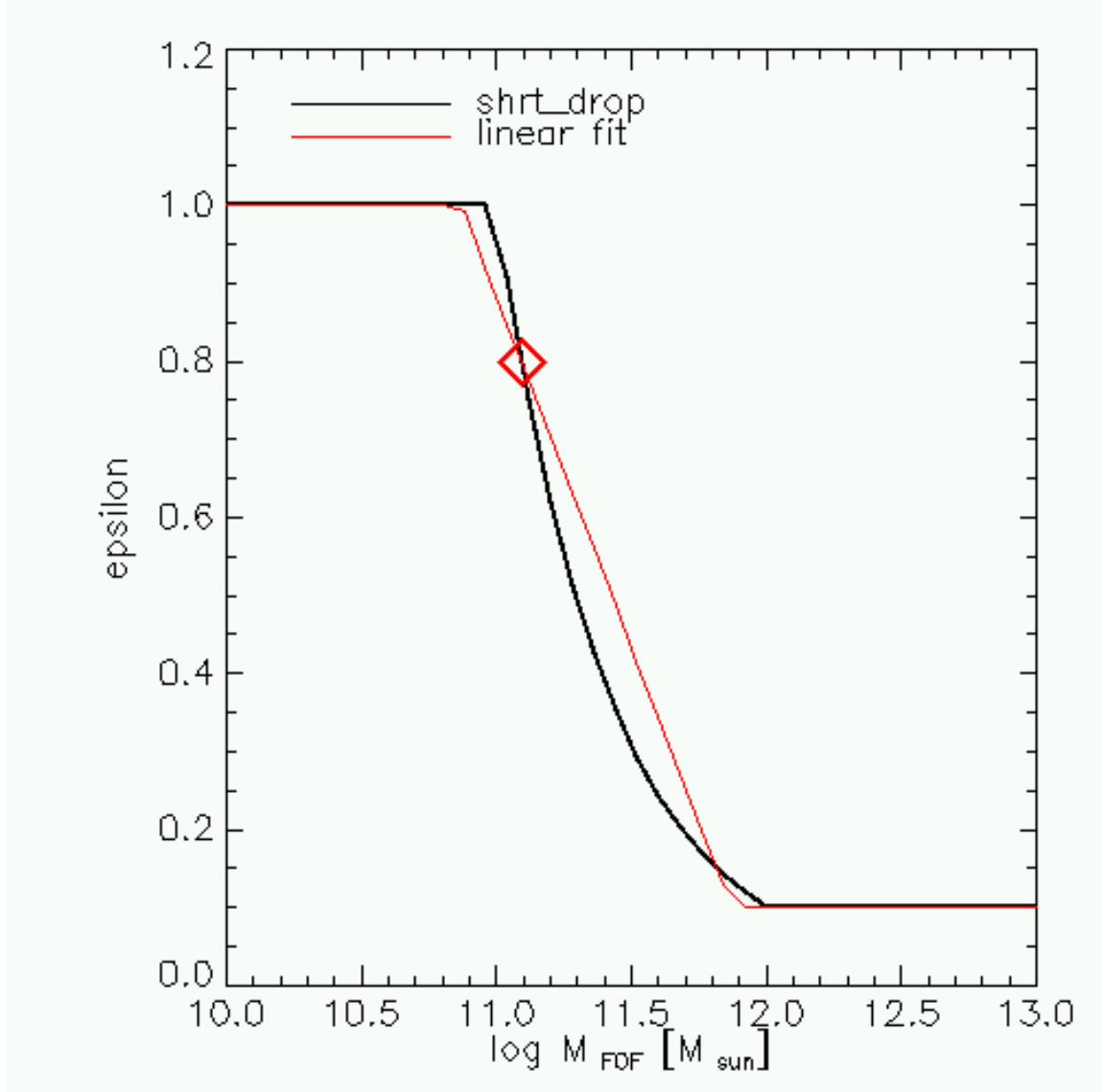


Figure 3.6: Used super nova energy as function of M_{FOF} for the short drop parameters (black) compared to a linear fit (red). The slope of the linear fit is determined such that the red line is parallel to the line through the points $(\epsilon_{\text{max}}, \log M_{\text{min}})$ and $(\epsilon_{\text{min}}, \log M_{\text{max}})$, where for example $\log M_{\text{min}}$ is the value of M_{FOF} below which $\epsilon = \epsilon_{\text{max}}$. The red fit has $\epsilon = 0.8$ for the same value of M_{FOF} as the short drop model.

thermal energy of the gas particle be M , \mathbf{V} , and U , and let m_i , \mathbf{v}_i and u_i be the same for the amount of accreted material, for $i = 1 \cdots N$.

Therefore conservation of mass & momentum yields

$$M_N = M + \sum m_i \quad (3.179)$$

$$M_N \mathbf{V}_N = M \mathbf{V} + \sum m_i \mathbf{v}_i \quad (3.180)$$

When a particle is accreted with specific energy u_i , we want to add $m_i u_i$ to the total thermal energy of the accretor. We can do this event by event, and still be independent of order (and hence number of cores, for example), by updating as follows

$$M_{i+1} = M_i + m_i \quad (3.181)$$

$$U_{i+1} = \frac{M_i U_i + m_i u_i}{M_{i+1}}, \quad (3.182)$$

then after all N events $M_N = M + \sum m_i$, and $M_N U_N = M U + \sum m_i u_i$, as required.

To conserve energy, we want

$$\frac{1}{2} M_N \mathbf{V}_N^2 + \Delta E = \frac{1}{2} M \mathbf{V}^2 + \sum_i \frac{1}{2} m_i \mathbf{v}_i^2, \quad (3.183)$$

where we should add the missing kinetic energy ΔE to the thermal energy of the final gas particle. For example the addition of one particle increases the thermal energy of the accreting particle by

$$\Delta E = \frac{1}{2} \frac{M m_1}{M + m_1} (\mathbf{V} - \mathbf{v}_1)^2 > 0. \quad (3.184)$$

Remember that in GADGET the velocity variable is $a^2 d\mathbf{x}/dt$ whereas the peculiar velocity (which enters in the equations above) is $a d\mathbf{x}/dt$, so the missing energy then contains a factor $1/a^2$.

We can also take into account the *thermal* energy of the accreted matter, which then leads to

$$\frac{1}{2} M_N \mathbf{V}_N^2 + \Delta E = \frac{1}{2} M \mathbf{V}^2 + \sum_i \frac{1}{2} m_i \mathbf{v}_i^2 + \sum_i m_i u_i. \quad (3.185)$$

Taking into account the dependence of the various quantities on the scale factor a , and denoting programme values as \hat{u} , say, then gives

$$\text{kinetic energy} = h^{-1} U_M U_V^2 a^2 \frac{1}{2} \sum \hat{m} \hat{v}^2, \quad (3.186)$$

and

$$\text{thermal energy} = h^{-1} U_M U_V^2 \sum \hat{m} \frac{\hat{S} (a^{-3} \hat{\rho})^{\gamma-1}}{\gamma - 1}. \quad (3.187)$$

In the code, we can update the momentum of the gas particle event by event, but we use an array $Energy[i]$ for each SPH particle to compute the right hand side in in Eq. (??). After having computed the new kinetic energy, $(1/2) M_N \mathbf{V}_N^2$, we can now compute the change in energy ΔE , and hence compute the change in entropy.

3.5.6 Black holes implementation

Basic equations

The Eddington luminosity of a BH with mass M_{BH} is

$$L_{\text{Edd}} = \frac{4\pi G M_{\text{BH}} m_p c}{\sigma_T} \quad (3.188)$$

where σ_T is the Thomson cross section. The Bondi-Hoyle-Littleton accretion rate in gas of density ρ and sound speed c_s , moving at speed v with respect to the BH, is

$$\dot{M}_{\text{acc}} = \alpha \frac{4\pi G^2 M^2 \rho}{(c_s^2 + v^2)^{3/2}}, \quad (3.189)$$

where α is a dimensionless efficiency parameter.

The increase in mass of the BH, \dot{M}_{BH} , and the radiative energy released, \dot{L} , are then related by

$$\dot{M}_{\text{BH}} = (1 - \epsilon_r) \dot{M}_{\text{acc}} \quad (3.190)$$

$$\dot{L} = \epsilon_r \dot{M}_{\text{acc}} c^2, \quad (3.191)$$

where $\epsilon_r \approx 0.1$ is the radiative efficiency.

The energy production \dot{L} is limited by the Eddington luminosity, so that

$$\dot{L} \leq L_{\text{Edd}} \quad (3.192)$$

$$\dot{M}_{\text{acc}} = \frac{\dot{M}_{\text{BH}}}{1 - \epsilon_r} \leq \frac{4\pi G M_{\text{BH}} m_p c}{\sigma_T \epsilon_r c^2}. \quad (3.193)$$

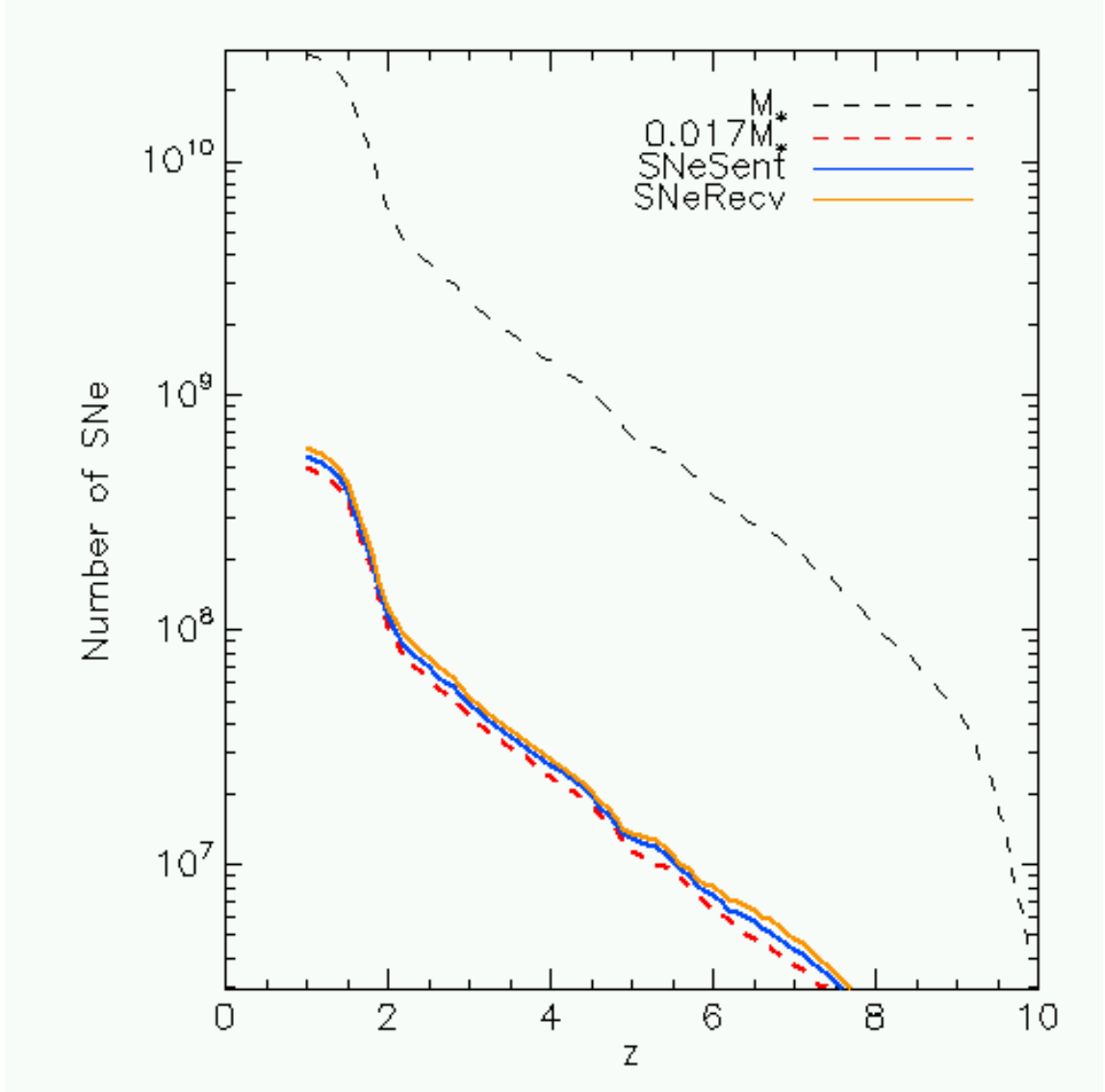


Figure 3.7: Total (initial) stellar mass as function of redshift (black dashed) and 0.017 times that (red dashed). A Chabrier IMF yields approx 0.017 SNe per solar mass, so the red dashed line is approximately the cumulative number of SNe. Blue solid line is the actual cumulative number of SNe that provided feedback in the code. Orange is the net cumulative number of SNe detected by the gas particles in the feedback routine. This is computed from the change in thermal energy of the gas particles. The blue and orange lines have been shifted upward by factors 1.1, and 1.2, for clarity: otherwise they both fall on top of the red dashed line. This demonstrates that the SN rate is consistent with the amount of stars formed, as well as that the gas receives the energy released.

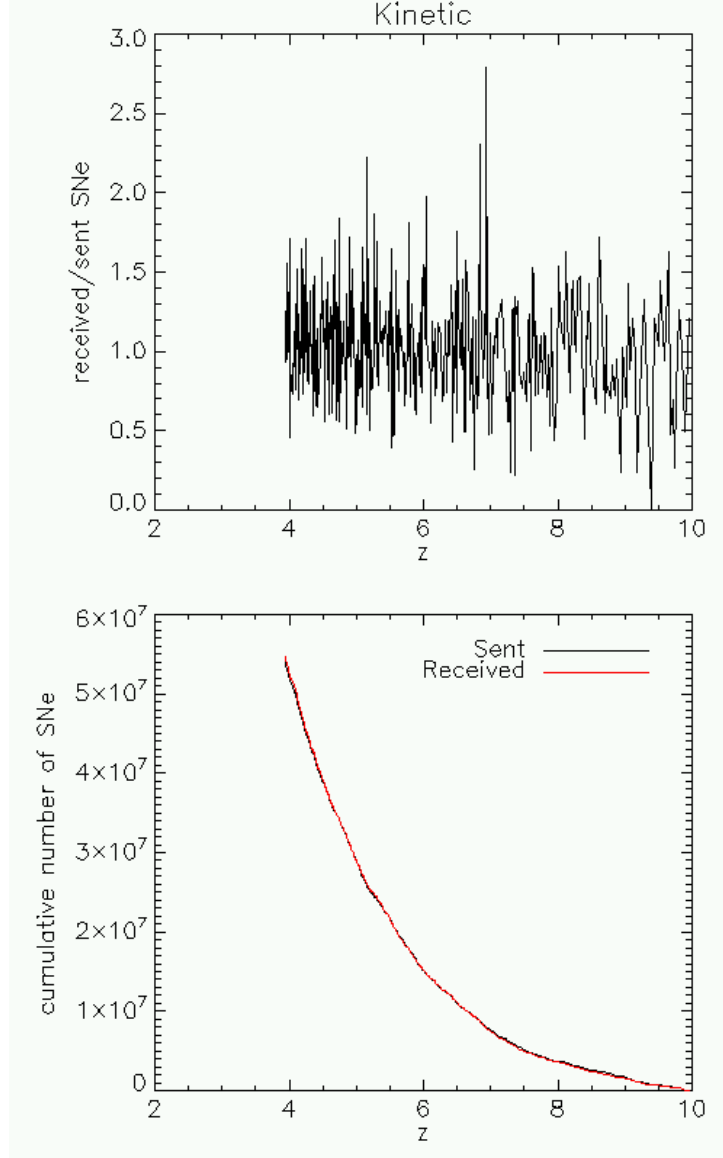


Figure 3.8: Top: ratio received/injected SN energy, for kinetic feedback. Bottom: cumulative number of injected SNe (black) versus energy received (red). Received energy is computed from the difference in total kinetic + thermal energy of the gas, between the end and the beginning of the feedback.

Numerical implementation and units

As usual we will use a hat to indicate a programme value.

$$M = U_M \hat{M} \quad (3.194)$$

$$v = U_V a^{-1} \hat{v} \quad (3.195)$$

$$c_s = a^{-3(\gamma-1)/2} U_v \hat{c}_s \quad (3.196)$$

$$\rho = a^{-3} U_M U_L^{-3} \hat{\rho} \quad (3.197)$$

$$G = U_L^3 U_M^{-1} U_T^{-2} \hat{G} \quad (3.198)$$

$$t = U_T \hat{t}. \quad (3.199)$$

Note that in Eagle, the units we use are set in the parameter file as

$$U_M = 10^{10} h^{-1} M_\odot \quad (3.200)$$

$$U_L = h^{-1} \text{Mpc} = 3.085 \times 10^{24} h^{-1} \text{cm} \quad (3.201)$$

$$U_V = \text{km s}^{-1} = 10^5 \text{cm s}^{-1} \quad (3.202)$$

and the unit of time is thus a derived unit,

$$U_T = U_L U_V^{-1} = 3.085 \times 10^{19} h^{-1} \text{s}, \quad (3.203)$$

which, just like the programme's gravitational constant, \hat{G} , is calculated in routine `begrn.c`. See the section on units for where the expansion factor a comes from, given our definition of co-moving variables. Note that \hat{v} is the programme's velocity (not the one in the output files), and the sound speed $\hat{c}_s^2 = \gamma \hat{p} \hat{\rho}^{-1} = \gamma \hat{S} \hat{\rho}^{\gamma-1}$, where S is the entropy.

Therefore we find that

$$\dot{M}_{\text{acc}} = \frac{U_M}{U_T} \propto \frac{4\pi \hat{G}^2 \hat{M}^2 \hat{\rho} a^3}{(a^{-3(\gamma-1)} \hat{c}_s^2 + a^{-2} \hat{v}^2)^{3/2}} \quad (3.204)$$

$$\equiv \frac{U_M}{U_T} \dot{\hat{M}}_{\text{acc}}. \quad (3.205)$$

This accretion rate is limited by the Eddington accretion rate, which is

$$\dot{M}_{\text{Edd}} = \frac{4\pi G M_{\text{BH}} m_{\text{p}} c}{\sigma_{\text{T}} \epsilon_r c^2} \quad (3.206)$$

$$= \frac{U_L^3 U_M^{-1} U_T^{-1} m_{\text{p}}}{\sigma_{\text{T}} c} \frac{U_M}{U_T} \frac{4\pi \hat{G} \hat{M}_{\text{BH}}}{\epsilon_r}. \quad (3.207)$$

Note that the first factor is a dimensionless constant, for a given choice of units.

Time variable. The following are gadget definitions:

$$\text{HUBBLE} \equiv 100 \text{ kms}^{-1} \text{ Mpc}^{-1} \quad (3.208)$$

$$\text{Hubble} \equiv \text{HUBBLE } U_T \quad (3.209)$$

$$E(z) \equiv (\Omega_m(1+z)^3 + \Omega_\Lambda + (1 - \Omega_m - \Omega_\Lambda)(1+z)^2)^{1/2} \quad (3.210)$$

$$\text{hubble_a} \equiv \text{Hubble} \times E(z) \quad (3.211)$$

and therefore

$$H(z) = h U_T^{-1} \text{hubble_a}. \quad (3.212)$$

The physical time step, related to a change $d \ln(a) = da/a$, is then

$$\Delta t \approx \frac{a}{\dot{a}} \frac{\Delta a}{a} \quad (3.213)$$

$$= \frac{1}{h U_T^{-1} \text{hubble_a}} d \ln a \quad (3.214)$$

$$\Delta \hat{t} = \frac{1}{h \text{hubble_a}} d \ln a \quad (3.215)$$

$$\Delta t = U_T \Delta \hat{t}. \quad (3.216)$$

Note that from earlier, $\delta t \equiv d \ln(a) = 2^{\text{bin}} \times \text{Timebase_interval}$, hence we also have

$$\Delta t = \left[U_T \frac{1}{h \text{hubble_a}} \right] \delta \hat{t}, \quad (3.217)$$

where the factor $\delta \hat{t} \equiv d \ln(a)$, and the factor in brackets is the variable `All.cf_Time_to_cgs` defined in `eagle_timestep.c`. Finally, the blackhole mass should be updated as

$$\Delta M_{\text{BH}} = U_M \Delta \hat{M}_{\text{BH}} \quad (3.218)$$

$$= \dot{M}_{\text{BH}} \Delta t \quad (3.219)$$

$$= (1 - \epsilon_r) \dot{M}_{\text{acc}} \Delta t \quad (3.220)$$

$$= \frac{U_M}{U_T} (1 - \epsilon_r) \dot{M}_{\text{acc}} \frac{d \ln(a)}{h U_T^{-1} \text{hubble_a}}. \quad (3.221)$$

svnversion 21680 In this version of the code, these equations are implemented

using the following variables:

$$\text{meddington} = \frac{h^{-1} U_T}{U_M} \dot{M}_{\text{Edd}} \quad (3.222)$$

$$\text{mdot} = \frac{U_T}{U_M} \dot{M}_{\text{acc}} \quad (3.223)$$

$$dt = h U_T^{-1} \Delta t. \quad (3.224)$$

The accretion rate mdot is ‘limited’ by the Eddington rate, as

$$\text{mdot} \leq \text{BlackHoleEddingtonFactor} \times \text{meddington}, \quad (3.225)$$

or $\dot{M}_{\text{acc}} \leq \text{BlackHoleEddingtonFactor} h^{-1} \dot{M}_{\text{Edd}}$. Note there is a stray Hubble parameter h here.

The mass of the black hole is updated as

$$\Delta \hat{M}_{\text{BH}} = (1 - \epsilon_r) \text{mdot} dt \quad (3.226)$$

$$= h U_M^{-1} ((1 - \epsilon_r) \dot{M}_{\text{acc}} \Delta t), \quad (3.227)$$

which also has a stray h in it.

versions after 21680 The variables are now changed to

$$\text{mdot_edd_over_mBH} = \frac{4\pi G m_p U_T}{\epsilon_r c \sigma_T} \quad (3.228)$$

$$\text{meddington} = \text{mdot_edd_over_mBH} \hat{M}_{\text{BH}} \quad (3.229)$$

$$= \frac{U_T}{U_M} \dot{M}_{\text{Edd}} \quad (3.230)$$

$$\text{mdot} = \frac{U_T}{U_M} \dot{M}_{\text{acc}} \quad (3.231)$$

$$dt = U_T^{-1} \Delta t. \quad (3.232)$$

To limit the rate of accretion to the Eddington rate is now simply $\text{mdot} \leq \text{meddington}$. The increase in black hole mass is $(1 - \epsilon_r) \text{mdot} \times dt$. The energy reservoir increases by

$$\text{BH_Energy} + = \epsilon_{\text{BlackholeFeedbackFactor}} \epsilon_r (\text{mdot} \times dt) \hat{c}^2, \quad (3.233)$$

where $\hat{c} = c/U_V$ is the speed of light in programme units. These BH variables are therefore all stored in programme units.

Feedback is implemented in routine `eagle_bhfeedback.c` as follows. The variable

$$\begin{aligned} \Delta u &= \text{BlackHoleMinHeatTemp} \\ &\times \text{temp_to_u_cgs_primordial} \end{aligned} \quad (3.234)$$

$$\text{critical_energy_per_unit_mass} = \frac{\Delta u}{U_V^2} \equiv \Delta \hat{u} \quad (3.235)$$

is the desired increase of specific energy of a gas particle near a BH, with Δu in cgs units, and the programme variable `critical_energy_per_unit_mass` the specific energy in programme units. The function `temp_to_u_cgs_primordial` returns the conversion from temperature to the thermal energy, using the mean molecular weight for primordial composition.

The code calculates the actual number of gas neighbours of a BH, as well as the total mass of those neighbours, without kernel weighing, as (**need to determine how h is set**)

$$\text{num_ngb} = \sum_j \quad (3.236)$$

$$\text{ngb_mass} = \sum_j m_j \quad (3.237)$$

$$\langle m \rangle \equiv \frac{\text{ngb_mass}}{\text{num_ngb}}. \quad (3.238)$$

The quantity $\langle m \rangle$ is the mean mass of a neighbouring gas particle – we have seen that gas particle masses may vary over orders of magnitude when stellar mass loss is included.

The BH builds-up an energy reservoir for feedback, (`BH_Energy` in Eq.??), and will only heat particles when it has sufficient energy to heat `BlackHoleNumberOfneighboursToHeat` (an input parameter) gas neighbours. Therefore feedback will only take place when

$$\begin{aligned} \text{BH_Energy} &\geq \text{critical_energy} \\ &= \text{BlackHoleNumberOfneighboursToHeat} \\ &\times \Delta \hat{u} \times \langle m \rangle. \end{aligned} \quad (3.239)$$

If this inequality is not satisfied, the BH does not perform heating, and its energy reservoir (`BH_Energy`) may increase over time.

If the BH does have sufficient energy in its reservoir, $\text{BH_Energy} \geq \text{critical_energy}$, we calculate the heating probability p per gas neighbour from

$$n_{\text{heat}} = \frac{\text{BH_Energy}}{\Delta\hat{u} \times \langle m \rangle} \quad (3.240)$$

$$p = \frac{n_{\text{heat}}}{\text{num_ngb}}. \quad (3.241)$$

If the BH heats few particles, $p < 1$ is the probability of heating a particle. However if the BH has a lot of energy, p can be larger than one. In that case, we put $p = 1$, and heat every particle to the higher specific energy

$$\Delta\hat{u} = \frac{\text{BH_Energy}}{\text{ngb_mass}}. \quad (3.242)$$

After a heating event, we zero the energy reservoir, BH_Energy .

Important: I changed the order of feedback and accretion. Up to version *svnversion 21680*, we first perform feedback, and then accretion. This has the unwanted effect that a heated particle may next be swallowed – cancelling out any feedback. The later versions first perform merging and accretion, and *then* perform feedback.

3.6 Time stepping

3.6.1 Tim-stepping: original scheme

Active particles: kick-drift-kick scheme

Assume velocity (v), position (x) and entropy (S) and predicted entropy (S_p) are known at time t . A time-step starts by computing a new current time-step Δt (see later for actual variables used), then steps the solution thorough the following routines

1. calculate new time-step Δt .
2. kick: kick *active* particles over 1/2 a time step: `do_first_halfstep_kick`

$$v^{n+1/2} = v^n + a^{n-1/2} \frac{\Delta t}{2} \quad (3.243)$$

$$S^{n+1/2} = S^n + \dot{S}^n \frac{\Delta t}{2}. \quad (3.244)$$

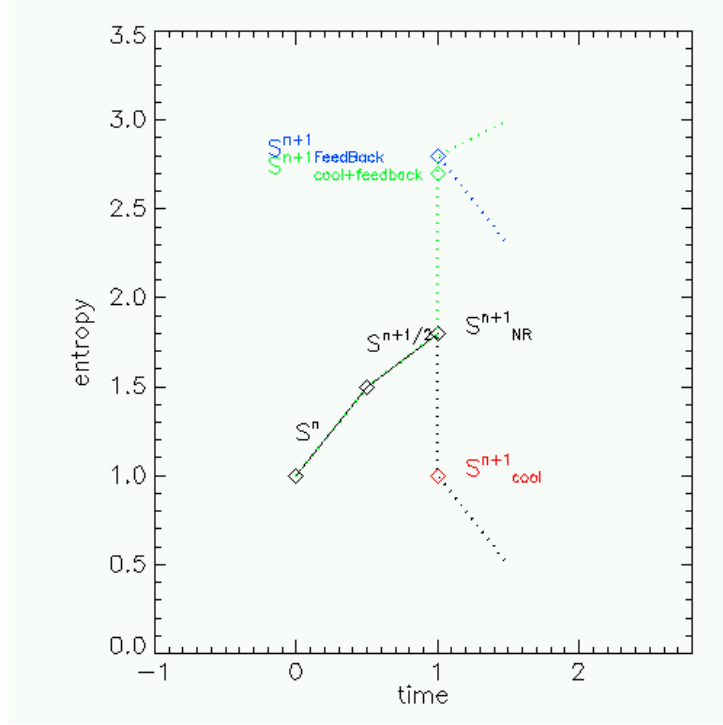


Figure 3.9: Different choices for selecting how to update entropy in the presence of radiative cooling and feedback. Black shows a case where the cooling rate is calculated comparing S^{n+1}_{NR} , the entropy calculated as $S^{n+1}_{NR} = S^{n+1/2} + \frac{dS^{n+1/2}}{dt}_{NR} \frac{\Delta t}{2}$, with S^{n+1}_{cool} , after an implicit cooling step, giving a cooling rate $\frac{dS^{n+1}}{dt}_{Cool} = (S^{n+1}_{cool} - S^{n+1}_{NR})/(\Delta t/2)$. After feedback, the entropy is much higher and the cooling rate should be much smaller. This is illustrated comparing blue dashed line (with the high cooling rate) against the green line, which uses the much smaller cooling rate computed at the higher entropy following feedback. The entropy after the next kick will now in fact increase (because of the low cooling rate) whereas in the old case it would decrease.

The entropy rate $\dot{S}^n = \dot{S}_{\text{NR}}^{n-1/2} + \dot{S}_{\text{cool}}^n$ combines the non-radiative rate of entropy change from time $t^{n-1/2}$ with the rate due to cooling, evaluated at time t^n .

3. drift: drift *all* particles to current time t : `find_next_sync_point_and_drift`

$$\Delta t_f = t - T_i^{\text{current}} \quad (3.245)$$

$$x(t) = x + v \Delta t_f \quad (3.246)$$

$$v_p(t) = v_p + a \Delta t_f \quad (3.247)$$

$$S_p(t) = S_p + \dot{S} \Delta t_f \quad (3.248)$$

$$T_i^{\text{current}} = t. \quad (3.249)$$

Note that T_i^{current} is simply the last time that the particle was drifted. Since Δt_f is calculated as the difference between the current time and the last time that the particle was drifted, we can simply add $v^{n+1/2} \Delta t_f$ and $\dot{S}^{n+1/2} \Delta t_f$ to get incremental updates to position, ‘predicted’ velocity v_p , and ‘predicted’ entropy S_p . The ‘accelerations’ v , a and \dot{S} correspond to the last velocity, accelerations, and entropy derivative computed.

4. calculate new accelerations: `compute_accelerations`, $a^{n+1/2}$ as well as the non-radiative rate of change of entropy, $\dot{S}_{\text{NR}}^{n+1/2}$. In `hydra.c`, accelerations are calculated using predicted values for velocity $v_p(t)$, and entropy $S_p(t)$, as well as the current ‘drifted’ particle positions $x(t)$. Note that the entropy here is a predicted value, and is store in a different variable than the entropy S used in the previous kick.
5. kick: kick *active* particles over second half of time step: `do_second_halfstep_kick`

$$v^{n+1} = v^{n+1/2} + a^{n+1/2} \frac{\Delta t}{2} \quad (3.250)$$

$$\hat{S}^{n+1} = S^{n+1/2} + \dot{S}_{\text{NR}}^{n+1/2} \frac{\Delta t}{2}. \quad (3.251)$$

Note in particular that this uses the non-radiative rate of change of entropy (*i.e.* ignores radiative heating/cooling in the entropy).

6. cooling: `eagle_docooling`. Calculate the cooling rate from

$$S^{\text{new}} = \text{do_cooling}(\hat{S}^{n+1}, \frac{\Delta t}{2}) \quad (3.252)$$

$$S^{\text{new}} \geq S_{\text{min}} \quad (3.253)$$

$$\dot{S}_{\text{R}}^{n+1} = \frac{S^{\text{new}} - \hat{S}^{n+1}}{\Delta t/2} \quad (3.254)$$

$$\dot{S}^{n+1} = \frac{S^{\text{new}} - S^{n+1/2}}{\Delta t/2} = \dot{S}_{\text{R}}^{n+1} + \dot{S}_{\text{NR}}^{n+1/2} \quad (3.255)$$

$$S^{n+1} = S^{\text{new}} \quad (3.256)$$

$$S_p^{n+1} = S^{n+1}. \quad (3.257)$$

This is I believe what we should do. However currently the time interval used in Eqs (??) and (??) is obtained from `eagle_get_SPHparticle_elapsed_time`, and (when the limiter is not used), equals Δt (and not $\Delta t/2$ as it should be). The rate of change of entropy, in Eq.(??) currently in the code is $\dot{S}^{n+1} = \dot{S}_R^{n+1}$, is *only* the radiave rate (which is why I claim we miss 1/2 a step of shock heating in the first kick).

Incidentally this illustrates why I claim our testing is not very rigorous: a Sod shock tube test would not show that we do not update the entropy correctly, because we would run it without cooling (as opposed to running it with zero cooling rate).

Passive particles

Passive particles need a predicted value of the entropy, S_p (for example used in `get_pressure`). Whereas active particles follow the kick-drift-kick scheme above, passive particles *only* get drifted. Since Δt_f is calculated as the difference between the current time and the last time that the particle was drifted, we can simply add $v^{n+1/2} \Delta t_f$ and $\dot{S}^{n+1/2} \Delta t_f$ to get incremental updates to position, velocity, and entropy.

3.6.2 Time-stepping: suggested scheme

A proper leap-frog scheme for entropy is not possible since \dot{S} is a function of S . For the non-radiative change (as computed in `hydro_force`), S is in fact not used: what is used is the pressure, which in routine `get_pressure` is calculated from the predicted entropy, S_p .

Also, since the radiative change in entropy is calculated using sub-cycling, associating a radiative rate of change of entropy from $\Delta S/\Delta t$ may give erroneous results. The scheme below is designed to be robust.

3.6.3 Cooling tests

The following tests do not use the timestep limiter.

Test 1: Interpolation of cooling tables

This test (`EAGLE_TEST_COOL=1`) simply compares the cooling rate (for a given density and redshift) for each element, as function of temperature, as interpolated by the code, against what is in the cooling tables. Code works well.

Test 3: cooling of box at uniform density and temperature

The uniform density box is initialized with a given density and temperature and then left to cool. The cooling rate is independent of temperature, and in this case the analytical solution is simply that T decreases linearly in time. In the code, cooling is switched off below $T = 10^4$ K. As seen from Fig. ??, the temperature evolution computed using EAGLE follows the analytical solution very well. Run this test setting `EAGLE_TEST_COOL=3`. This tests the implementation of routine that evolves the thermal energy, for a case where there are no adiabatic changes, nor non-adiabatic changes that are due to shocks.

Test 4: cooling through a Sod shock

This non-cosmological set-up is a shock tube, and is run with `EAGLE_TEST_COOL=4`. The `CoolingTest/SodCool` directory contains a python script (`analytic.py`) that calculates the similarity solution discussed below, and an idl script to compare the simulation to that solution, as well as the `Config.sh` and initial conditions files. See Creasy+ '11 for details.

Initially, gas in a long tube is of uniform density and temperature. The gas is given a velocity, so that gas at $x < 0$ has $v_x > 0$ and gas at $x > 0$ has $v_x < 0$: this generates a strong shock at $x = 0$ which propagates out. Figure ?? compares the analytic solution to the EAGLE solution when the

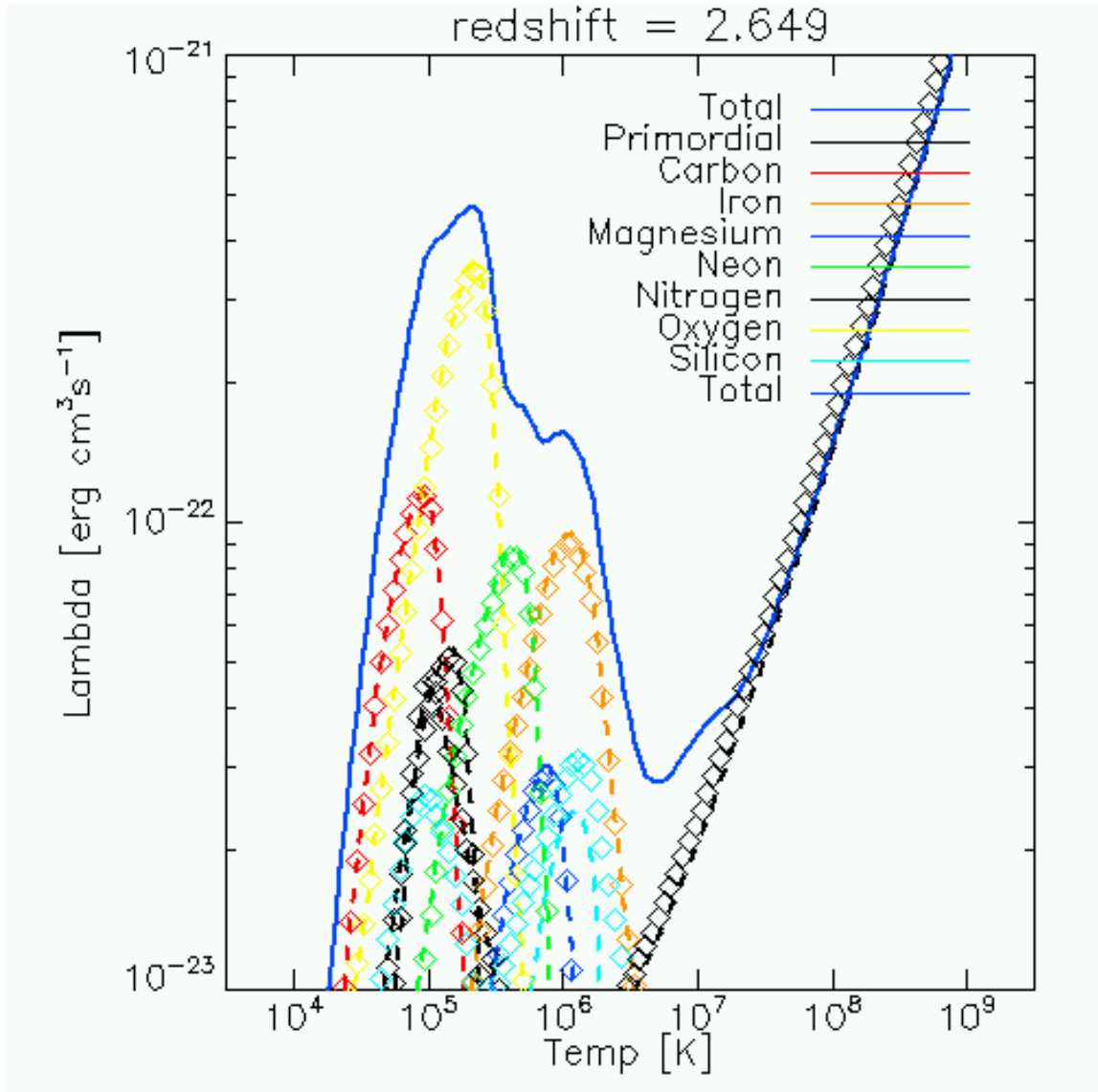


Figure 3.10: Comparison of cooling rate at $z = 2$. for gas at a given density, between values calculated by the code (dashed lines) and rates directly from the cooling table.

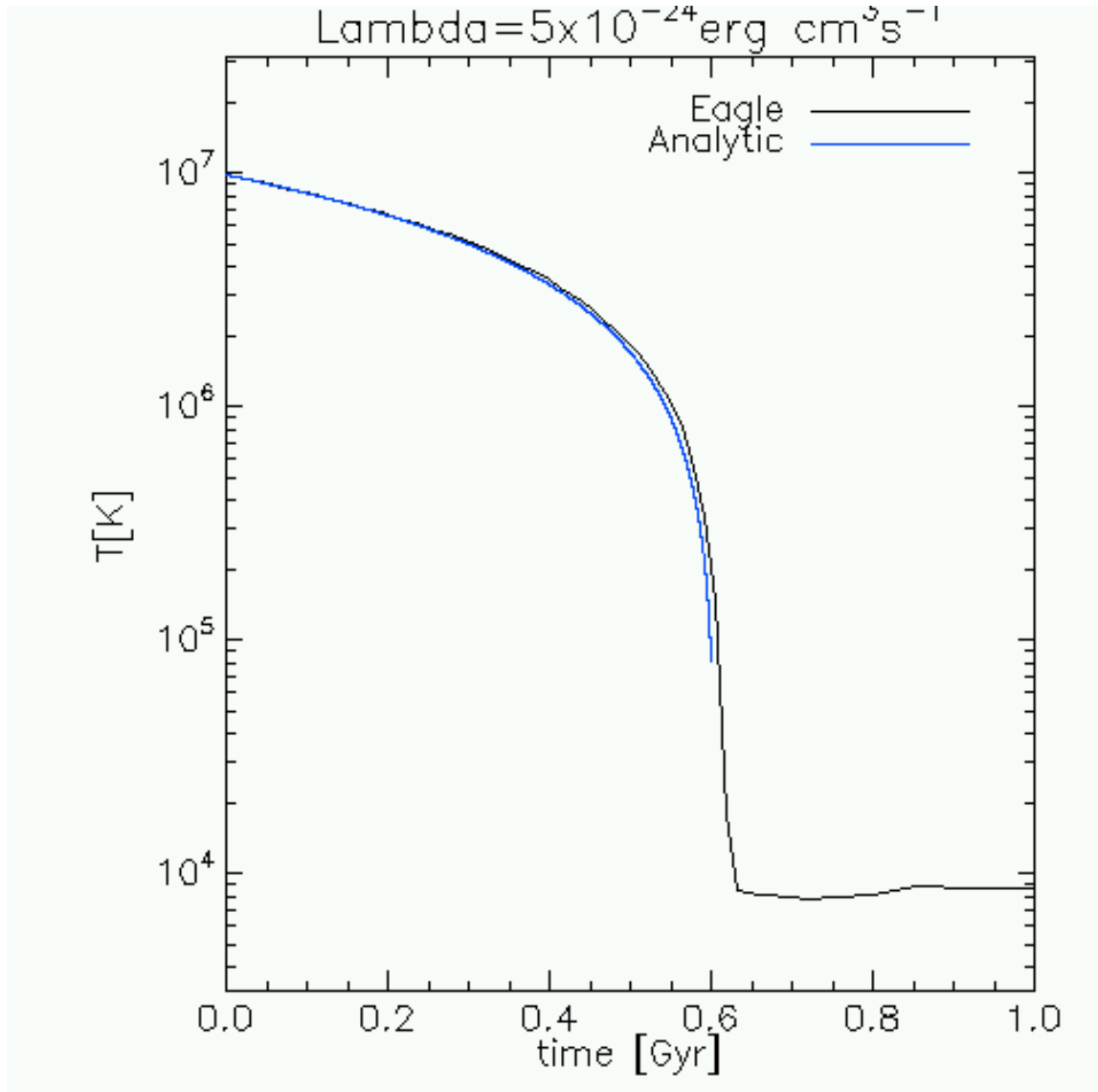


Figure 3.11: Non-cosmological cooling of gas at uniform density and temperature, with a rate which is independent of T . The analytical solution (gas cools linearly in time) and the EAGLE solution track each other well. In the code, gas stops cooling below $T = 10^4$ K.

gas does not cool ($\gamma = 5/3$). Apart from a wall heating effect at $x = 0$, the code reproduces the similarity solution well.

Figure ?? is the case with cooling included. Again EAGLE seems to do OK. This tests the combination of non-adiabatic (shock) and strong entropy changes of the code.

Test 5: evolution of temperature at the mean density in cosmological volume

Test of the evolution $T(z)$ at the mean density are shown in Fig. ?. In a test where cooling and reionization heating is applied to a gas element at the mean density (black curve), the resulting $T(z)$ does as expected, and is close to the data points of Schaye+00. Blue line is $u(z)$ scaled to match $T(z)$ at the maximum value: the blue line then follows the black line expect at high $z > 6$ where the Universe is not yet very highly ionised (and hence the mean molecular weight is still changing). Red line is the (scale) cumulative injected thermal energy (due to H and He reionization). These curves look fine.

However the red solid circles correspond to a simulation with low σ_8 , and corresponds to the median temperature: clearly there is an issue here: those points should lie on the black line.

3.7 Comparison to previous code

Appendix:

Variables, their meaning, units, and scaling

This note describes the units and meaning of variables in the EAGLE version of GADGET, both for internal code units, and those that are written to the snapshot files, when a cosmological simulation is performed (i.e., when the code uses comoving variables).

The Hubble constant is written as $H_0(z = 0) = 100 h \text{ km s}^{-1}$, and $a \equiv 1/(z + 1)$ denotes the expansion factor, with z the redshift. The subsection name refers to the group name in the hdf5 file

We will denote code variables by a hat, so \hat{x} is the physical comoving position (which has dimensions), whereas \hat{x} is the corresponding code position

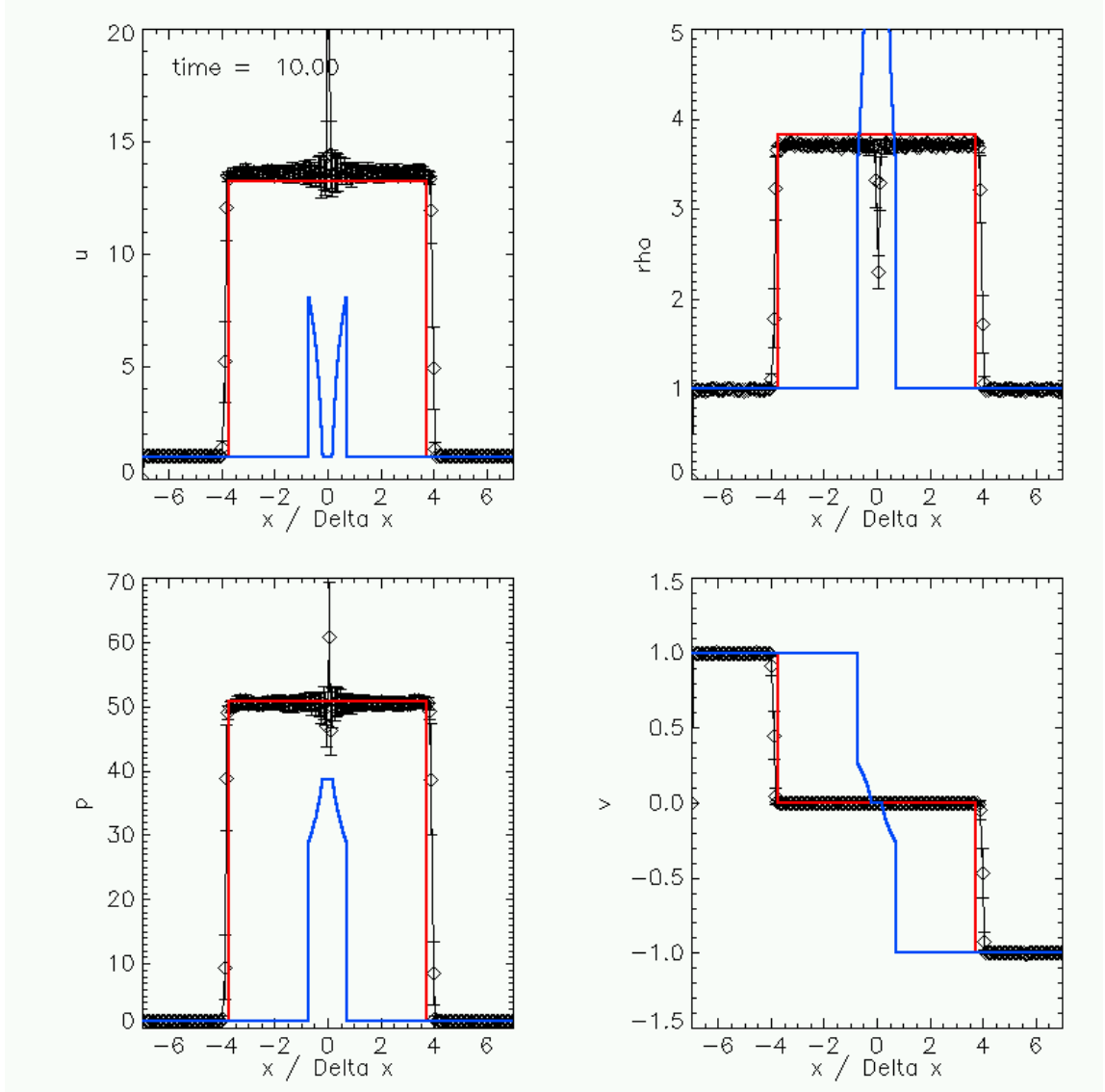


Figure 3.12: Non-cooling Sod shock: uniform density and temperature gas from the left shocks when it runs into the gas that comes from the right. Apart from wall-heating at $x = 0$, EAGLE gets the solution (red line) right. The black data points are median SPH values in bins.

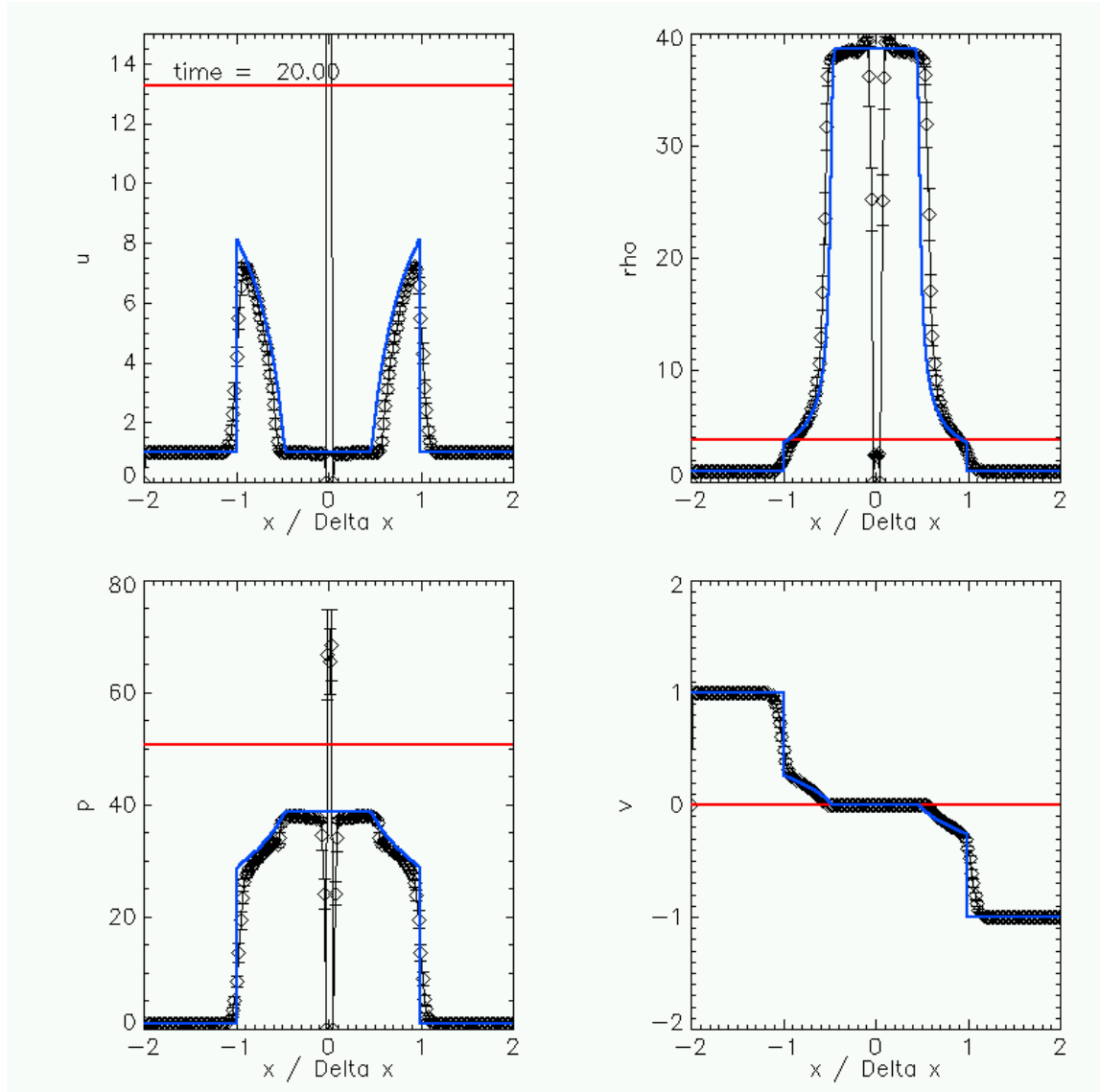


Figure 3.13: Same as Fig.?? but including cooling, the blue line is the similarity solution.

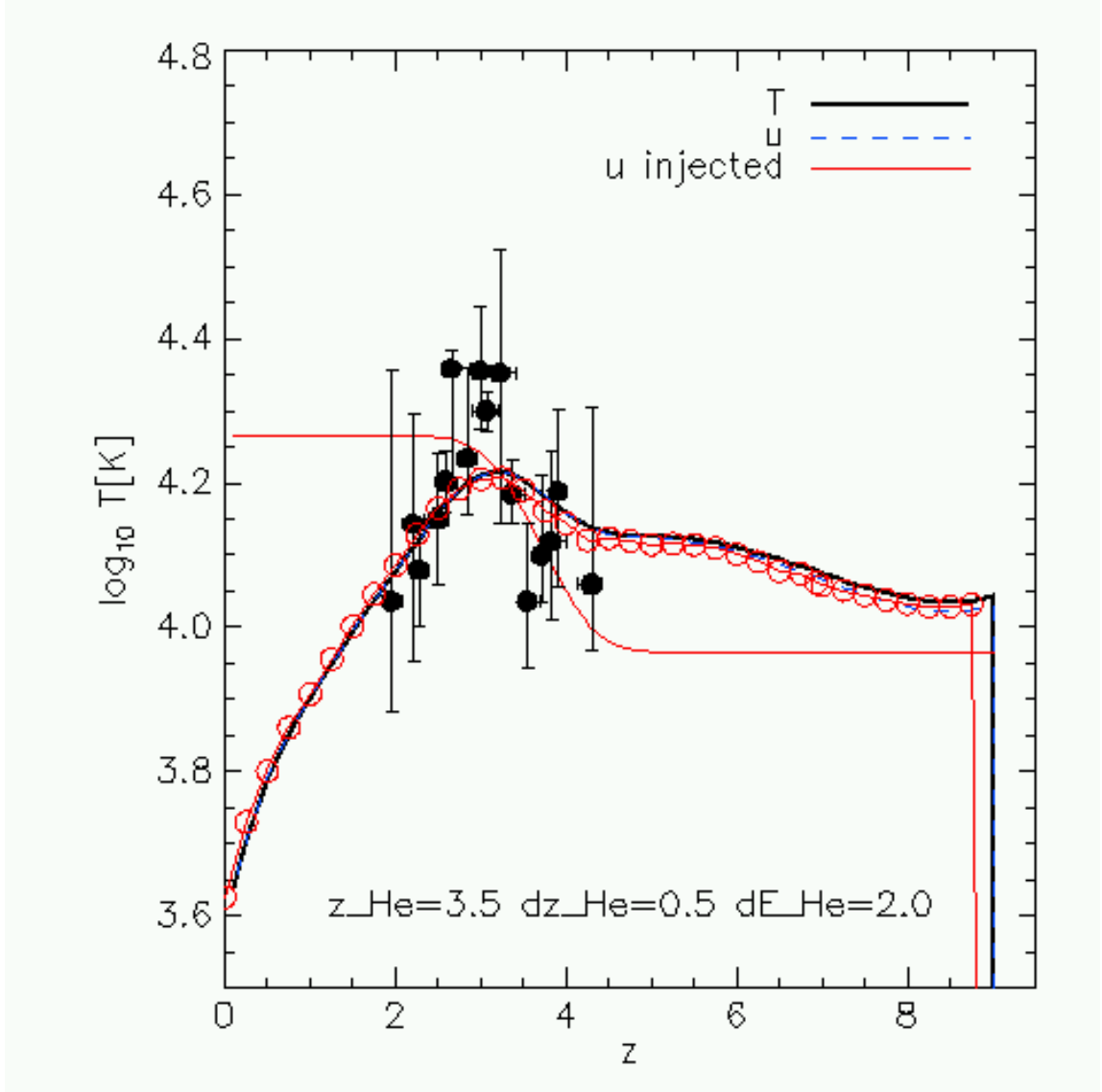


Figure 3.14: Evolution of T at the mean density. *Red* curve is proportional to the internal energy injected due to hydrogen and helium reionization. *Black* (*blue*) curve is evolution of $T(z)$ ($u(z)$), using a test routine that calls EAGLECOOLINGRATE to evolve the thermal state of a parcel of gas at the mean density. Black points with error bars are those of Schaye+00. *Red points* are the $T(z)$ of a simulation with very low σ_8 : this tests the cosmological update of the cooling rates: the code clearly passes this test.

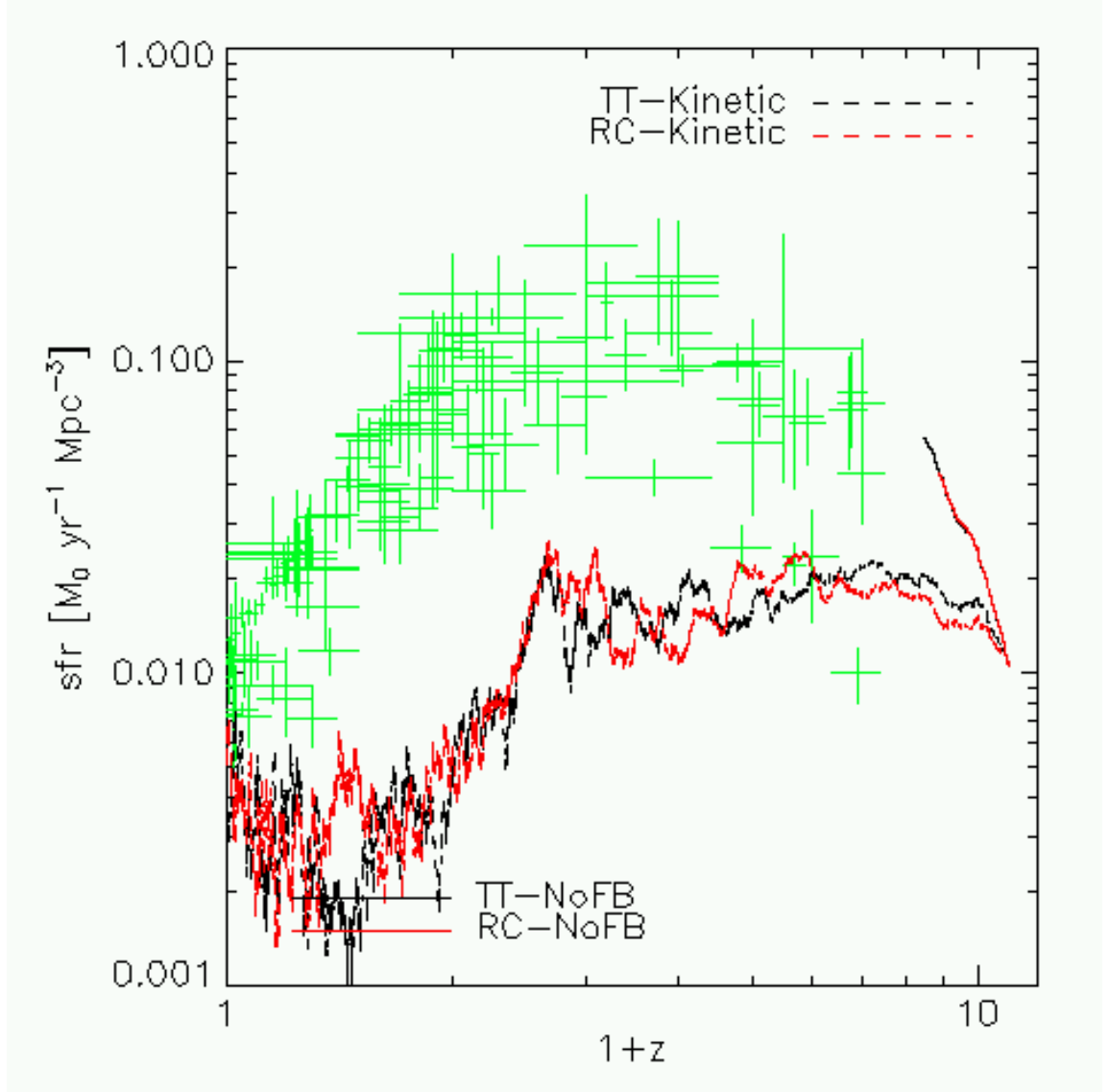


Figure 3.15: Madau plot for runs without feedback (solid) and runs with constant wind speed kinetic feedback ($v_w = 647 \text{ km s}^{-1}$, $\dot{M}_w/\dot{M}_{\star} = 4.17$) (dotted) between original code (red) and new implementation (black). Box is $3.125 h^{-1} \text{ Mpc}$, 64^3 particles.

Table 3.1: Translation between input parameters and physical parameters for star formation

process	symbol	name input variable	default
Star formation law	A_{KS}	SF_SchmidtLawCoeff_MSUNpYRpKPC	$1.5151 \times 10^{-4} M_{\odot} \text{ yr}^{-1} \text{ kpc}^{-2}$
	n_{KS}	SF_SchmidtLawExponent	1.4
	$\rho_{\star, \text{KS}}$	SF_SchmidtLawHighDensThresh_HpCM	$10^3 \text{ hydrogen cm}^{-3}$
	n'_{KS}	SF_SchmidtLawHighDensExponent	2
Cool-eos	ρ_{eos}	EOS_Cool_MinPhysDens_HpCM3	hydrogen particles 0.1 c
	Δ_{eos}	EOS_Cool_MinOverDens	10
	γ_{eos}	EOS_Cool_GammaEffective	1
	T_{eos}	EOS_Cool_TempNorm_K	8000K
Jeans-eos	ρ_{eos}	EOS_Jeans_MinPhysDens_HpCM	hydrogen particles 10 c
	Δ_{eos}	EOS_Jeans_MinOverDens	10
	γ_{eos}	EOS_Jeans_GammaEffective	4/3
	T_{eos}	EOS_Jeans_TempNorm_K	8000K

(which is dimensionless). Note that the variable in the output file, x_{out} , can be different from either x or \hat{x} .

Definitions

The relation between physical and comoving variables is defined using a as

$$r \equiv ax . \quad (3.258)$$

where r is the physical position, and x the comoving one. This is used to define the peculiar velocity, v_p , and peculiar acceleration, a_p , as

$$\begin{aligned} v_p &\equiv a \frac{dx}{dt} \\ a_p &\equiv \frac{1}{a} \frac{d}{dt} (a v_p) . \end{aligned} \quad (3.259)$$

Let p and ρ be the physical pressure, and density, respectively. The Eulerian equations of motion are

$$\ddot{\mathbf{x}} + 2H\dot{\mathbf{x}} + (\dot{\mathbf{x}} \cdot \nabla)\dot{\mathbf{x}} = -\frac{1}{a^3}\nabla\Psi - \frac{1}{a^2}\frac{\nabla p}{\rho}$$

$$\nabla^2\Psi = 4\pi G\rho_{g,0}\delta, \quad (3.260)$$

where ∇ is d/dx , δ is the overdensity, $\rho_{g,0}$ is the mean physical density at $a = 1$, and $H = \dot{a}/a$. We will assume a polytropic equation of state, $p = \epsilon \rho^\gamma$, where ϵ is the entropy.

Units

The Units group defines the conversion from GADGET dimensionless variables into cgs units. These are read from the parameter file and copied into the hdf5 file. At present these are

1. Length: $U_L = \text{UnitLength_in_cm}$
2. Mass: $U_M = \text{UnitMass_in_g}$
3. Velocity: $U_V = \text{UnitVelocity_in_cm_per_s}$
4. Energy: $U_L \equiv U_M U_V^2$.

For later use, the derived density unit $U_\rho \equiv U_M U_L^{-3}$ pressure unit $U_p = U_D U_V^2$, entropy unit, $\epsilon = p/\rho^\gamma$, $U_\epsilon = U_p U_\rho^{-\gamma}$, and time unit, $U_t = U_L U_V^{-1}$.

Constants

The units of these mathematical and physical constants are cgs. Solar abundance $Z_{\text{sol}}^{\text{ar}}$ is the assumed solar metallicity as a mass fraction.

Header

1. Time: expansion factor a (dimensionless), with $a = 1$ today.
2. Redshift $= z = 1/a - 1$
3. HubbleParam $= h$
4. NumPartTotal: 5 integers: total number of particle of each species.

Time, and Hubble constant

The parameter file defines h , the dimensionless Hubble constant. Several places in the code compute

$$\mathcal{H} = 100 \text{ km s}^{-1} \text{ Mpc}^{-1} U_t \left(\Omega_m a^{-3} + \Omega_\Lambda + (1 - \Omega_m - \Omega_\Lambda) a^{-2} \right)^{1/2}, \quad (3.261)$$

so that $H(t) = h \mathcal{H} U_t^{-1} = \dot{a}/a$ is the Hubble constant. This can be used to compute the timestep,

$$\frac{\Delta t}{h^{-1} U_t} = \frac{\Delta a}{a} \frac{1}{\mathcal{H}(a)} = \Delta \hat{t}. \quad (3.262)$$

Note that GADGET uses the expansion factor a as time variable, but in these notes I will always use t to denote time, never expansion factor.

Particle variables

N is the number of particles. There are five particle species, PartType0 is gas, PartType4 are stars, the others are dark matter. The number of particles of each species can be determined from the Header variable NumPartTotal.

3.8 Gadget equations

1. The name of the snapshot variable that contains the particle mass is Masses(1:N):

$$\begin{aligned} m &= \text{Masses } h^{-1} U_M \text{ [g]} \\ &= \hat{m} h^{-1} U_M \text{ [g]}. \end{aligned} \quad (3.263)$$

2. The name of the snapshot variable that contains the particle position is Coordinates(3,N):

$$\begin{aligned} \mathbf{r} &\equiv a \mathbf{x} \\ &= \text{Coordinates} \times h^{-1} a U_L \text{ [cm]} \\ &= \hat{\mathbf{x}} \times h^{-1} a U_L \text{ [cm]} \end{aligned} \quad (3.264)$$

3. The name of the snapshot variable that contains the particle velocities is `Velocities(3,N)`:

$$\begin{aligned}\mathbf{v}_p &\equiv a \dot{\mathbf{x}} \\ &= \text{Velocities} \times a^{1/2} U_V \text{ [cm/s]} .\end{aligned}\tag{3.265}$$

The internal velocity $\hat{\mathbf{v}} = a^2 h^{-1} \frac{d\hat{\mathbf{x}}}{dt}$

4. The name of the snapshot variable that contains the particle accelerations is `Acceleration(3,N)`:

$$\begin{aligned}\mathbf{a}_p &\equiv \frac{1}{a} \frac{d}{dt} a \mathbf{v}_p \\ &= a \ddot{\mathbf{x}} + 2H(a) a \dot{\mathbf{x}} \\ &= \text{Acceleration} h U_V^2 U_L^{-1} \text{ [cm/s}^2\text{]} .\end{aligned}\tag{3.266}$$

The internal acceleration $\hat{\mathbf{a}}$ differs for hydro and gravity. For gravity:

$$\begin{aligned}\hat{\mathbf{a}}_{\text{grav}} &= \sum \hat{G} \frac{\hat{m} \hat{\mathbf{x}}}{\hat{x}^3} \\ &= a \frac{d\hat{\mathbf{v}}}{dt} .\end{aligned}\tag{3.267}$$

whereas for hydro

$$\begin{aligned}\hat{\mathbf{a}}_{\text{hydro}} &= \frac{1}{\hat{\rho}} \frac{d\hat{p}}{d\hat{\mathbf{x}}} \\ &= a^{3(\gamma-1)} \frac{d\hat{\mathbf{v}}}{dt} .\end{aligned}\tag{3.268}$$

5. The name of the snapshot variable that contains the particle density is `Density`:

$$\begin{aligned}\rho &= \text{Density} h^2 a^{-3} U_M U_L^{-3} \text{ [g cm}^{-3}\text{]} \\ &= \hat{\rho} a^{-3} h^2 U_\rho \text{ [gcm}^{-3}\text{]} .\end{aligned}\tag{3.269}$$

6. `GADGET` does not output the entropy nor the pressure, but it is useful to know what the internal values are.

$$\begin{aligned}
p &= \hat{p} a^{-3\gamma} h^2 U_p \quad [\text{g} (\text{cm s}^{-1})^2 \text{cm}^{-3}] \\
\epsilon &= \hat{\epsilon} h^{2-2\gamma} U_p U_\rho^{-\gamma} \quad [\text{g} (\text{cm s}^{-1})^2 \text{cm}^{-3} (\text{g cm}^{-3})^{-\gamma}]. \quad (3.270)
\end{aligned}$$

7. The name of the variable that contains the particle internal energy per unit mass, u , is InternalEnergy.

$$\begin{aligned}
u &= \text{InternalEnergy} U_V^2 \quad [(\text{cm/s})^2] \\
&= \hat{u} U_V^2 \quad [(\text{cm/s})^2]. \quad (3.271)
\end{aligned}$$

The internal energy u is usually computed from the entropy, $u = \epsilon \rho^{\gamma-1}/(\gamma - 1)$, hence during a runs one needs to first convert the co-moving density to a physical value, $\rho \propto \hat{\rho} a^{-3}$ which then introduces a factor $a^{-3(\gamma-1)}$ since the entropy ϵ is a independent.

The temperature is computed from $u = k_{\text{Boltz}} T/(\gamma - 1) \mu m_H$, where k_{Boltz} is Boltzmann's constant, μ the mean molecular weight per particle, and m_H is 1/12 of the mass of a ^{12}C Carbon nucleus.

8. The name of the variable that contains the particle potential is Potential(1:N):

$$\begin{aligned}
\Phi &\equiv \sum \frac{GM}{r} \\
&= \text{Potential} a^{-1} U_V^2 \quad [(\text{cm/s})^2]. \quad (3.272)
\end{aligned}$$

9. OnEquationOfState.

Has to do with being on the effective equation of state (EoS), namely

$= 0$ when particle has never been on EoS
 $= 1$ when particle is currently on EoS
 $= -a$ when particle not currently on EoS, but left EoS when expansion factor was a .

10. IronFronSNIa Mass fraction of Iron (Iron mass over particle mass), produced solely by super novae of type I.
11. StarFormationRate. The star formation rate (if a gas particle), or the star formation rate of the gas particle when the star formed, for a star particle.
12. TimeMaximumEntropy, TimeMaximumTemperature. Expansion factor when particle had its maximum entropy, and maximum temperature, respectively. (Note that temperature and entropy do not scale with a .)
13. WindFlag.

$= 0$ if particle has never been in a wind
 $= -a$ when particle was last in the wind at expansion factor a

14. MaximumEntropy(1:N):

Maximum value of the entropy $\hat{\epsilon}$. Physical entropy

$$\epsilon_{\max} = \text{MaximumEntropy} h^{2-2\gamma} U_{\epsilon} [\text{g (cms}^{-1}\text{)}^2 \text{cm}^{-3} (\text{g cm}^{-3})^{-\gamma}]. \quad (3.274)$$

Star formation implementation

Write the gas surface density, Σ_H , in terms of the hydrogen column density, N_H , and hydrogen mass fraction X_H , as

$$\Sigma_{\text{gas}} = N_H \frac{m_H}{X_H} = \left(\frac{p\gamma f}{G} \right)^{1/2}, \quad (3.275)$$

where p is the effective pressure.

The Schmidt law is written as

$$\frac{\Sigma_{\text{SFR}}}{M_{\odot} \text{ kpc}^{-2} \text{ yr}^{-1}} = C \left(\frac{\Sigma_{\text{gas}}}{M_{\odot} \text{ pc}^{-2}} \right)^n \quad (3.276)$$

Combining the last two equations gives

$$\frac{\Sigma_{\text{SFR}}}{M_{\odot} \text{ kpc}^{-2} \text{ yr}^{-1}} = \mathcal{C} \left(\frac{p\gamma f}{G(M_{\odot} \text{ pc}^{-2})^2} \right)^{n/2}. \quad (3.277)$$

If you also assume that

$$\frac{\Sigma_{\text{SFR}}}{\Sigma_{\text{gas}}} = \frac{\dot{\rho}_{\star}}{\rho_{\text{gas}}}, \quad (3.278)$$

then

$$\begin{aligned} \dot{\rho}_{\star} &= \mathcal{C} \rho_{\text{gas}} \frac{\Sigma_{\text{SFR}}}{\Sigma_{\text{gas}}} \\ &= \mathcal{C} \frac{\rho_{\text{gas}}}{M_{\text{yr}}} \left(\frac{\Sigma_{\text{gas}}}{M_{\odot} \text{ pc}^{-2}} \right)^{n-1} \\ &= \mathcal{C} \frac{\rho_{\text{gas}}}{M_{\text{yr}}} \left(\frac{p\gamma f}{G(M_{\odot} \text{ pc}^{-2})^2} \right)^{(n-1)/2}. \end{aligned} \quad (3.279)$$

In terms of numerical implementation this becomes

$$\frac{dm_{\star}}{dt} = \mathcal{C} \frac{m_{\text{gas}}}{M_{\text{yr}}} \left(\frac{p\gamma f}{G(M_{\odot} \text{ pc}^{-2})^2} \right)^{(n-1)/2} \quad (3.280)$$

or in terms of programme variables

$$\frac{d\hat{m}_{\star}}{d\hat{t}} = \mathcal{C} \hat{m}_{\text{gas}} \frac{h^{-1} U_t}{10^6 \text{ yr}/s} \left(\frac{\hat{p} a^{-3\gamma} h^2 \gamma f}{\text{GRAVITY}} \left(\frac{(\text{pc/cm})^2}{M_{\odot}/\text{g}} \right)^2 \right)^{(n-1)/2}. \quad (3.281)$$

where the quantity in $()$ is dimensionless, and $\text{GRAVITY} = 6.67 \times 10^{-8}$ is Newton's constant in cgs units.