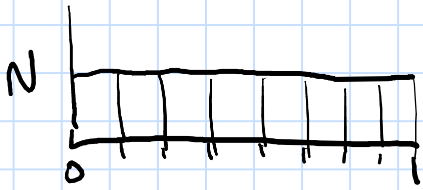
Setting up initial conditions with random number generators

- Generator produces numbers uniformly between 0 and 1

  e.g. 0.1, 0.9, 0.6, 0.4, 0.1, 0.3 ... etc



- In Python: numpy random default_rng(seed=none)

- Random number seed: if chosen, generator will produce the same series of numbers each time

---

Example: Use of random number generator to produce uniform density cube of particles

```
Do i = 1, Npart
    x = ran (seed)
    y = ran (seed)
    z = ran (seed)
    Print, x, y, z
Enddo
```

Placing particles on spherical shells:

```
$r_s = 1$ Kpc                                   Shell radius

Do $i = 1$, Npart
    $X_1$ = ran(seed)                           Select two random
    $X_2$ = ran(seed)                           numbers
    $z = (1.0 - 2.0\, X_1)\, r_s$               Z from -1 to +1
    $x = (r_s^2 - z^2)^{1/2} \cos_{rad}(2\pi X_2)$   Ensures
    $y = (r_s^2 - z^2)^{1/2} \sin(2\pi X_2)$         $r^2 = x^2 + y^2 + z^2$
End do
```

Letting the radius of the shell vary (max radius=1 kpc):

```
$r_{max} = 1$ Kpc                               Maximum radius
Do $i = 1$, Npart
    $X_1$ = ran(seed)                           Select three
    $X_2$ = ran(seed)                           random numbers
    $X_3$ = ran(seed)
    $r_s = X_3 \times r_{max}$                  r varies uniformly from 0
                                                to 1 for each particle
    $z = (1.0 - 2.0\, X_1)\, r_s$
    $x = (r_s^2 - z^2)^{1/2} \cos_{rad}(2\pi X_2)$   X Y Z selected
    $y = (r_s^2 - z^2)^{1/2} \sin(2\pi X_2)$         as before
End do
```

Plummer sphere, model for star clusters:

Density Profile: Plummer

$$\rho(r) = \underbrace{\frac{3M_{pl}}{4\pi r_{pl}^3}}_{\rho_0}\left(1+\frac{r^2}{r_{pl}^2}\right)^{-5/2}$$

Enclosed mass:

$$M(<r) = \frac{M_{pl}\, r^3}{(r^2+r_{pl}^2)^{1.5}}$$

M_{pl} :	Total mass	Key parameters
r_{pl} :	scale radius of plummer	

Using enclosed mass to set up density distribution:

- First rearrange M(<r) equation for r:

$$r = r_{pl}\left[\left(\frac{M(<r)}{M_{pl}}\right)^{-2/3} - 1\right]^{-1/2}$$

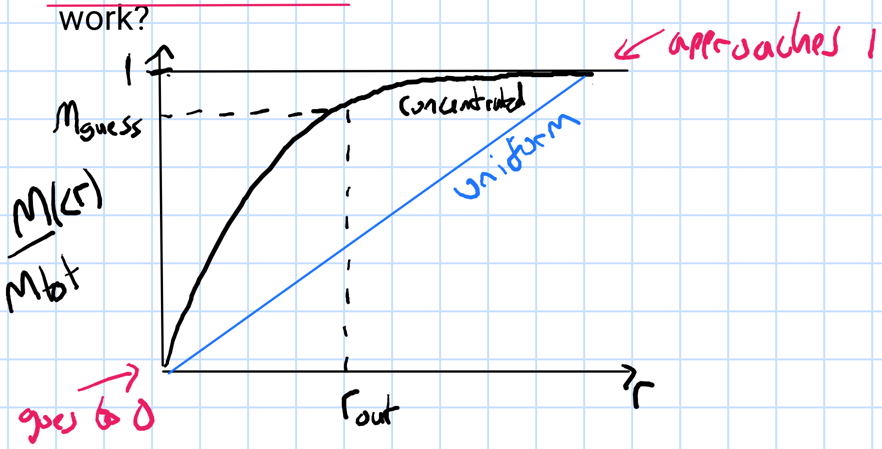
- Choose random number for M(<r)/Mpl

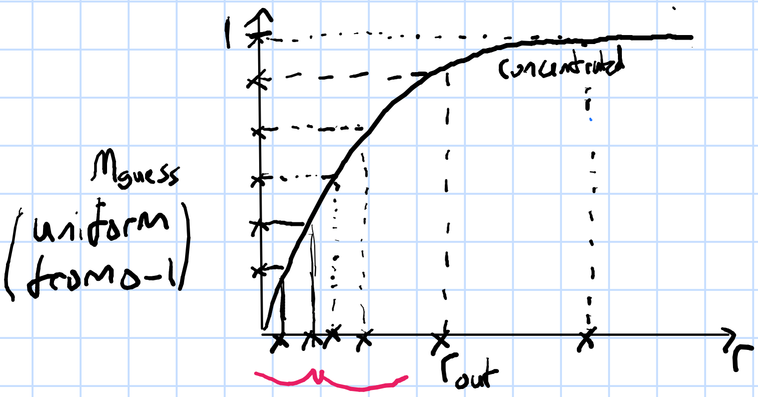
```
Do  i = 1, Npart
        X_1 = ran(seed)
        r = r_pl (X_i^{-2/3} - 1)^{-1/2}
Enddo.
```

- Resulting radial distribution will naturally follow a plummer sphere density distribution

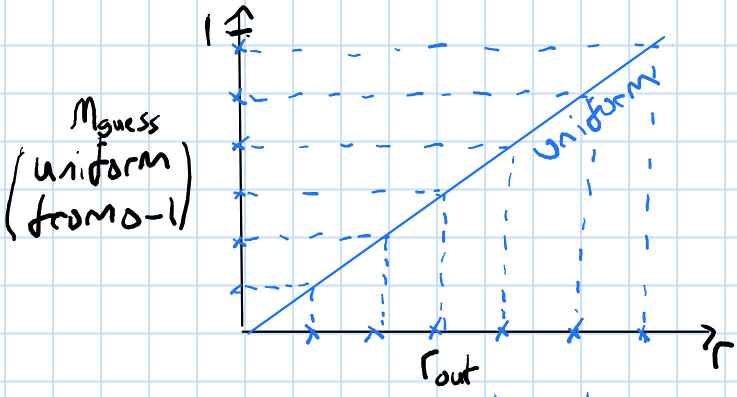
  Useful check: $r_{half} = 1.3\, r_{pl}$ rhalf is radius containing half the total mass

How does it work?





r_{out} prefers smaller values



r_{out} is uniform too

Setting particle velocities using the probability distribution function of escape velocities:

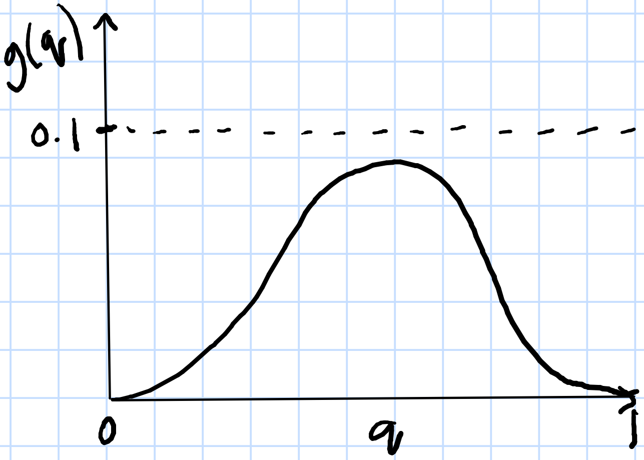
- Escape velocity of plummer:

$$V_{esc} = \left(\frac{2GM_{pl}}{r_{pl}}\right)^{1/2} \left(1 + \frac{r^2}{r_{pl}^2}\right)^{-1/4}$$

- Let $q = V/V_{esc}$ (q from 0–1)

- Probability distribution function for plummer sphere:

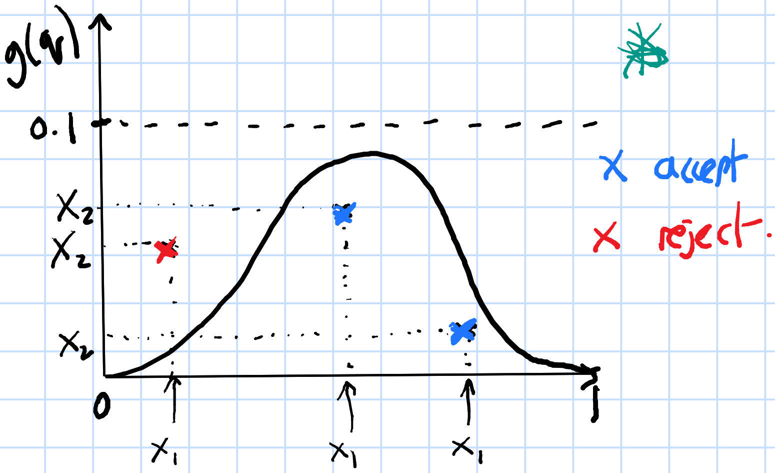
$$g(q) = q^2(1-q^2)^{3.5}$$



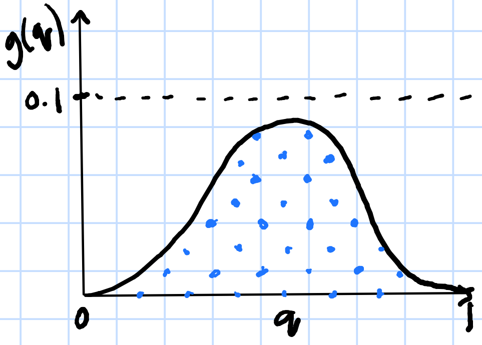
Usage

:

```
Do i = 1, Npart
    $X_1$ = ran(seed)
    $X_2$ = ran(seed)
    If $0.1 X_2 < \underbrace{X_1^2 (1 - X_1^2)^{3.5}}_{g(q)=q^2(1-q^2)^{3.5}}$ *
        ↳ yes, accept it ($V_{tot} = X_1 \cdot V_{esc}$)
        ↳ no, reject it, try again.
Enddo
```



For large Npart, we will uniformly sample under the curve



For an isotropic velocity field:

```
Do i = 1, Npart
    $x_1 = \mathrm{ran}(iseed)$
    $x_2 = \mathrm{ran}(iseed)$
    $V_z = (1.0 - 2.0 x_1) V_{tot}$
    $V_x = (V_{tot}^2 - V_z^2)^{1/2} \cos(2\pi x_2)$
    $V_y = (V_{tot}^2 - V_z^2)^{1/2} \sin(2\pi x_2)$
Enddo
```

The plummer model has an isotropic velocity field,

i.e., $\sigma_x = \sigma_y = \sigma_z$

What you must do:

Part 1: Write a code to set up your own Plummer sphere

(i) For Mpl=1000 Msol, and rpl=1.0 pc, set up the positions of a plummer sphere with with Npart=1000. Positions should be in units of parsec. From your particle distribution, measure the half mass radius and check if it is as expected for a plummer sphere

(ii) Now use the probability density function of Vesc to give your plummer sphere particles velocities (in units of km/s). Plot Vtot versus radius for each particle and overplot the curve of Vesc to check if the velocities are as expected.

A simple N-body code in python: nbody.py

Three parts:

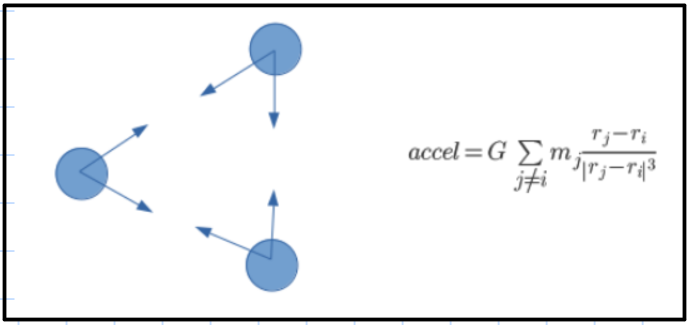
1) def getacc( ) | calculates accelerations

2) def getenergy( ) | Kinetic and Potential energy of Particles

3) def main ( ) | Main routine, getacc( ) and getenergy( ) called from main( )

How an N-body code works:

- Accelerations:



$$accel = G \sum_{j \neq i} m_j \frac{r_j - r_i}{|r_j - r_i|^3}$$

For each particle, calculate accelerations from all neighbouring particles

- Softening:

$$a = -\frac{GM}{r^2} \quad \therefore \quad r \to 0, \quad a \to \infty$$

solution: $$sep = \left(x^2 + y^2 + z^2 + l_{soft}^2\right)^{1/2}$$

Useful rule: lsoft is typically set to be ~ mean particle separation

- A leapfrog integration scheme:

A first calculation of the acceleration must be made before entering the time loop

```
For i in range (Nt)
    $V_i = V_i + \frac{\Delta t}{2} \times a_i$        ½ Update velocities
    $r_i = r_i + \Delta t \times V_i$                 full update positions
    Calculate accelerations
    $V_i = V_i + \frac{\Delta t}{2} \times a_i$        ½ Update velocities
    $t = t + \Delta t$                               Update time
```

Velocities updated in two steps, before and after positions are updated

Results are more stable and accurate than if velocities updated in one go at same time as positions

---

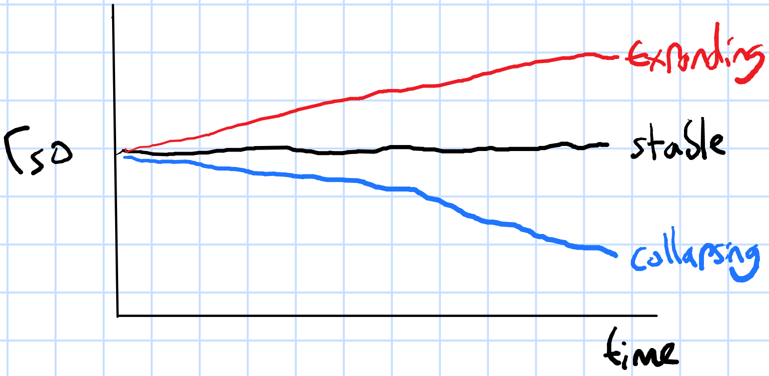
Important parameters in main ( ):

Note, nbody.py is a G=1 code

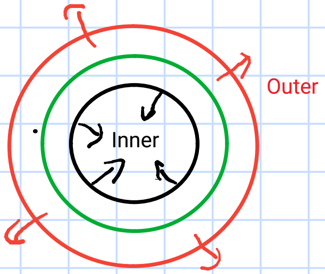
- N (# of particles)
- t (start time in G=1 units)
- tend (finish time in G=1 units)
- dt (time step in G=1 units)
- softening (softening length in G=1 units)
- Initial conditions (positions, velocities, masses) read from ascii input file (in G=1 units)

Testing model stability with Lagrangian radii:

- Half mass radius, r50: radius containing half the total mass



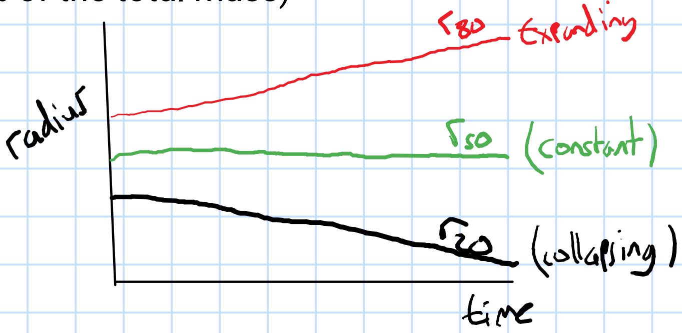
- But model could collapse in centre and expand in outskirts



shows evolution of inner regions

- For this we also need r20, and r80 (radius containing inner 20% & inner 80% of the total mass)

evolution of outer regions



- Can increase number of lines for even more detail: (e.g. r10, r25, r50, r75, r90, etc). These are called Lagrangian

A simple way to measure the Lagrangian radii for equal mass particles:

1) order the particles by radius from small down to large (Inside code using a function, or using the linux command 'shuf -n')

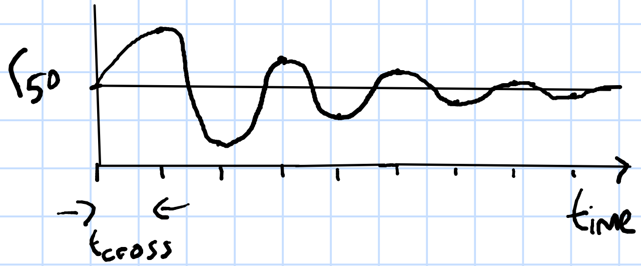
2) count until you find 10% of the rows (r10), then 20% of the rows (r20), etc

Note, if particles are different masses, need to do step 1 and measure cumulative mass at each radius

---

Relaxation timescales:

- A slightly out of equilibrium model will oscillate (expand & collapse) on timescales ~ the crossing time of the system.



System takes ~10 crossing times to fully relax

- tcross: time for typical particle to cross the scaleradius of the system
  - tcross for a Plummer sphere:

$$t_{cross} = \left(\frac{3\pi}{32}\right)^{-3/2}\left(\frac{r_{pl}^3}{GM_{pl}}\right)^{1/2}$$

What you must do:

Part 2: Test the stability of your plummer sphere from part I on the python N-body code:

(i) use G=1 units, with Lsim=1 pc, and Msim= 1 Msol. What is Tsim (in Myr) and Vsim (in m/s) in this unit system? Convert your initial conditions from part 1 into these units for input to the N-body code.

(ii) we will need to simulate the cluster for 10 crossing times to test its stability. Calculate what 10 crossing-times is in Myr and convert to Tsim units and use this value for the 'tend' parameter. Choose the time step 'dt' to ensure there are at least 50 steps per crossing time for accurate modelling.

(iii) given the half mass radius of a plummer sphere (see part 1), choose the softening length to be ~ the mean interparticle spacing.

(iv) run the simulation, and test the stability of the model using your own code to measure the Lagrangian radii.

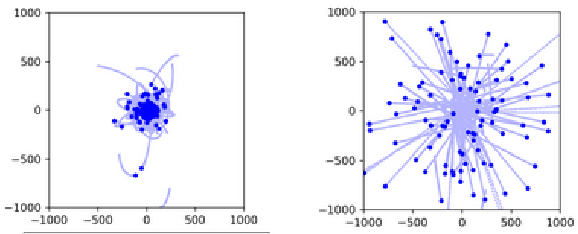
Embedded star clusters:

Stars form in clusters that are embedded in molecular gas

The first stars that go supernova can easily remove the gas from low mass clusters

The gas removal can have a dramatic effect on the stars in the cluster. Many young clusters are completely destroyed: "Infant mortality"

Infant mortality explains why we observe so many young clusters but much less older clusters

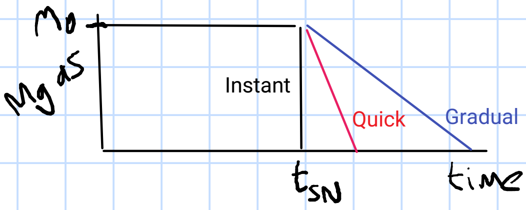


Movement of stars in star cluster (left) before gas loss and (right) after gas loss

Important parameters controlling destructiveness:

1) the gas fraction: $F_{gas} = \frac{M_{gas}}{M_{total}}$

2) how suddenly the gas is removed after the supernova time (t_SN):



Timescale usually measured in units of crossing times

What you must do:

Part 3: Model embedded star clusters & response to gas expulsion:

(i) by differentiating the potential of a plummer sphere, derive the acceleration components (ax, ay, az) that a plummer sphere of gas applies to embedded star particles.

(ii) identify in the N-body code where to include the potential from a gas plummer sphere and modify the code to allow it.

Note 1: for a leapfrog scheme, the acceleration must be calculated once before entering the timestep loop, and then for each timestep loop

Note 2: The total plummer sphere mass is unaffected by the gas fraction parameter (fgas). Instead, fgas controls how much of the total plummer sphere mass is in gas or stars. For simplicity, the total number of star particles is kept the same while lowering their individual masses to match the requested gas fraction.

E.g.,. If Mtot=1000 Msol, and fgas=0.5
then Mgas=500 Msol,
and M_starparticle is half original value (with no gas)

Identify where in the code fgas plays a role and explain how it is applied.

(iii) Using Lagrangian radii, test model stability with fgas=0.5 and fgas=0.95 (heavily gas dominated)

(iv) Alter the code to include gas removal that begins after 5 crossing times (t_SN=5 tcross). Model this so the gas mass decreases linearly after t_SN. Use a free parameter (tgone) for how long it takes all the gas to disappear (in units of tcross).

(v) Use Lagrangian radii plots to judge the survival of the star cluster to gas expulsion. Demonstrate the dependency on chosen fgas and tgone values. In which areas of parameter space are clusters unaffected and in which are they completely destroyed?