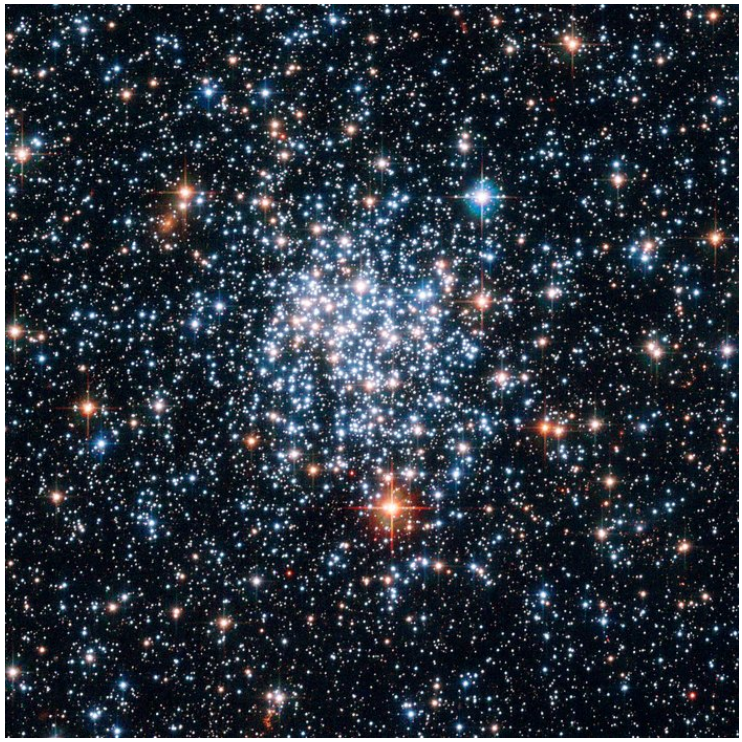
FYS4150: Project 5
N-body simulation of an Open Galactic Cluster

Andri Spilker and Tiffany Chamandy

December 5th, 2016



NGC 265, Hubble Space Telescope [4]

Contents

Abstract

This project utilises a simple N-body simulation to look at how an open galactic cluster evolves with time. The star particles interact by Newtonian gravity, and the time integration is done using the Velocity Verlet method. We study the system as time goes towards and past the associated collapse time τ_{crunch}, and find that the system does not collapse as is predicted for a completely uniform distribution, but rather stabilises at a Virial radius. Our study shows that on average 16% of the initial particles are shot out of the system, and that the Virial radius as well as the parameters in the density distribution of particles are strongly dependent on the initial number of particles N. This means that one cannot simply use a large number of particles to simulate a uniform distribution, and special care has to be taken to compute and understand such systems.

1 Introduction

In this project we make an N-body simulation of an open galactic cluster and look at how it evolves with Newtonian gravity over time. An open galactic cluster is a group of stars, bound together by gravity. The group is formed from the collapse of the same molecular cloud, and thus the stars have similar compositions and ages. The main difference between the stars are their mass. Open galactic clusters are therefore interesting to study, as they can teach us about stellar evolution. An example of an open galactic cluster can be seen on the front page, where we have included a Hubble Space Telescope image of NGC 265.

This project is based on the article 'Cold uniform spherical collapse revisited' by Joyce, Marcos and Labini from 2010 [5]. We use the Velocity Verlet method to study the time evolution of a cluster where each star is affected by the gravitational pull from the others. Open galactic clusters have short lifetimes (on cosmological scale) of only a few hundred million years, and we want to see what happens to such clusters when they disappear. It is especially interesting to see if the system collapses, if it reaches an equilibrium situation and if its energy is conserved. We follow the system over a few τ_{crunch}; the associated dynamical time scale of the system. This time is also called the free fall time, and is the time such a system would take to collapse to its center. The derivation of τ_{crunch} comes from spherical collapse, and is done below.

1.1 Theoretical background

We assume a perfectly spherical overdensity, and Gauss' law then says that all matter outside the sphere can be ignored. Newton's laws then give:

$$\frac{d^2r}{dt^2} = \frac{-GM}{r^2} \tag{1}$$

Integrating once we get

$$\dot{r}^2 = \frac{2GM}{r} + C \tag{2}$$

This ordinary differential equation has a parametrised solution:

$$R = A(1 - cos\theta) \tag{3}$$

$$t = B(\theta - sin\theta) \tag{4}$$

$$A^3 = GMB^2 \tag{5}$$

Where R is the radius of the spherical overdensity, t is the time, θ is a time variable and $C = A^2/B^2$ from the integration constant. A graphical illustration of what is happening in such a spherical collapse can be seen in figure 1.

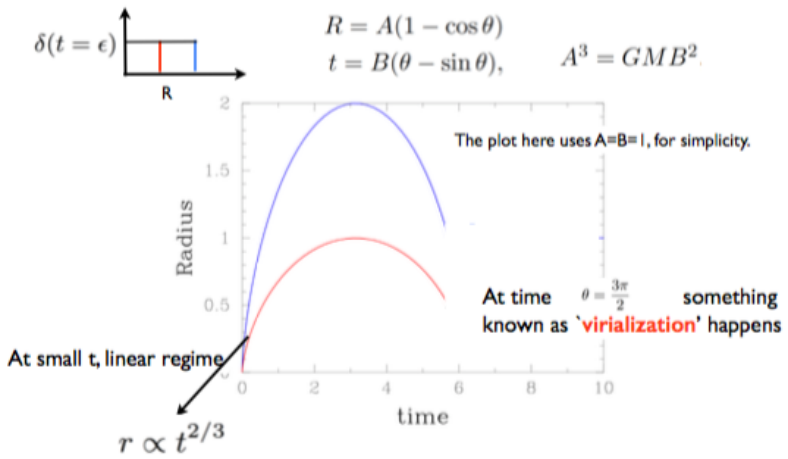


Figure 1: A graph showing the evolution of a spherical density perturbation with time. The figure is taken from the lecture notes of Cosmology and Extragalactic Astrophysics at UiO [3].

In this figure the perturbation starts out increasing in radius with the expansion of the universe, before gravity takes over and the perturbation starts to collapse. The time of Virialisation is indicated on the graph at $\theta = \frac{3\pi}{2}$ at a radius $R_{vir} = A = \frac{R_0}{2}$, while complete collapse happens at $\theta = 2\pi$ which corresponds to a time $t = 2\pi B$. The time

it takes to collapse is then the time from the peak of the graph (where the radius is R_0 and $t = \pi B$) to $t = 2\pi B$, so the collapse time is πB. Calculating this further gives:

$$t_{collapse} = \tau_{crunch} = \pi B = \pi \frac{A^{3/2}}{\sqrt{GM}} = 2\pi \frac{R_{vir}^{3/2}}{\sqrt{G\rho_0 V}} = \pi \frac{R_{vir}^{3/2}}{\sqrt{G\rho_0 \frac{4}{3}\pi R_0^3}} \tag{6}$$

Where $R_0 = 2R_{vir}$ (R_0 is at the peak of the graph), so that

$$\tau_{crunch} = \pi \frac{R_{vir}^{3/2}}{\sqrt{G\rho_0 \frac{4}{3}\pi 8 R_{vir}^3}} = \frac{\pi}{\sqrt{G\rho_0 \frac{4}{3}\pi 8}} = \sqrt{\frac{\pi^2}{G\rho_0 \frac{4}{3}\pi 8}} = \sqrt{\frac{3\pi}{32G\rho_0}} \tag{7}$$

Where ρ_0 is the initial density of the spherical over density. By inserting the density of our open cluster (mass of $1000M_{sun}$ and radius of 20 light years), we get that the collapse time ($=\tau_{crunch}$) is approximately 8 million years.

The aim of this project is to examine what happens to the system as the time reaches τ_{crunch}, and as it passes this time. We simulate a 'cold' uniform spherical collapse, so we start our star particles at rest, with zero velocity, in a spherical uniform (random) distribution. The timesteps are calculated using the Velocity Verlet method, which is described in section 2.1. We start with 100 star particles of masses in a Gaussian distribution around 10 solar masses with a standard deviation of 1 solar mass. These are positioned randomly within a sphere of radius $R_0 = 20ly$. We use τ_{crunch} as our unit of time, and the equations are made dimensionless by setting $\tau_{crunch} = 1$, so that the gravitational constant G becomes $G = \frac{\pi^2 R_0^3}{8N\mu\tau_{crunch}^2} = \frac{\pi^2 R_0^3}{8N\mu}$ with units of $\frac{ly^3}{y^2 M_{sun}}$, cancelling the units of the other parameters. The calculation of G is done below.

$$\tau_{crunch} = \sqrt{\frac{3\pi}{32G\rho_0}} \tag{8}$$

$$G = \frac{3\pi}{32\tau_{crunch}^2 \rho_0} = \frac{3\pi}{32\tau_{crunch}^2 \frac{M_{tot}}{V}} = \frac{3\pi}{32\tau_{crunch}^2 \frac{N\mu}{\frac{4}{3}\pi R_0^3}} = \frac{\pi^2 R_0^3}{8N\mu\tau_{crunch}^2} \tag{9}$$

Here R_0 is the initial radius of the spherical distribution (20 ly), N is the number of star particles in the system, and μ is the average mass of these star particles.

Later in the project (section 3.4) we increase the number of particles and decrease the mass. This is done in order to look at more continuous distributions of matter and how they behave under gravity. In the calculation of τ_{crunch} we assumed a uniform spherical over density, which is not really true when we only have 100 star particles. But as $N \to \infty$ we move towards a continuous fluid. Such approximations are often

used to represent baryonic and dark matter in simulations of the early universe. N-body simulations are often used in Cosmology to learn about structure formation in the early universe. To study structure formation, which is one of the main goals of cosmology, one needs to consider the collapse of gas clouds or dark matter halos. Such collapses must have occurred to produce the galaxies, black holes, stars and planets that we observe today. It is therefore very important to understand how N-body simulations work and what they can be used for, which is the goal of this project. Unfortunately we were not able to simulate more than 1000 particles in this project, as we ran out of time, and computing power. However, this report is still useful for understanding the basics of N-body simulations, the dependency on N of the results and its wide applications.

The program for this report was based on our earlier work on a simulation of the Solar system, and the report for the Solar system project can therefore be helpful to understand the construction of the classes and programs written, and can be found at [1].

2 Methods

All programs produced in this project can be found at the github address. The programming for this report was done in C++, all figures were made in Python, while the animations were made in Ovito.

https://github.com/akspilke/AstroProject5.

2.1 The Verlet Method

The equations necessary for solving the coupled ordinary differential equations with the Verlet method are solved for using taylor expansions (see report for Project 3 for derivation [1]) and come to a final result of:

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}a_i + O(h^3) \tag{10}$$

$$v_{i+1} = v_i + \frac{h}{2}(a_{i+1} + a_i) + O(h^3). \tag{11}$$

The factor a_{i+1} was not entirely trivial to implement into the code. Therefore we needed an intermediate step to solve for this factor. We did this by calculating the velocity at $i + \frac{1}{2}$ (for half time steps).

$$v_{i+\frac{1}{2}} = v_i + \frac{h}{2}a_i \tag{12}$$

$$x_{i+1} = x_i + hv_{i+\frac{1}{2}} \tag{13}$$

The forces of the system then needed to be calculated again before we calculated a_{i+1}, because the acceleration and force are coupled. This step is shown as:

$$v_{i+\frac{1}{2}+\frac{1}{2}} = v_{i+1} = v_{i+\frac{1}{2}} + \frac{h}{2}a_{i+1}. \tag{14}$$

We have included the Verlet Class from our C++ script below. This class solves for the position and velocity evolution with time, using the methods just described, see figure 2.

```cpp
#include "verlet.h"
#include "solarsystem.h"

Verlet::Verlet(double dt) :
    m_dt(dt)
{

}

void Verlet::integrateOneStep(SolarSystem &system)
{
    system.calculateForcesAndEnergy();

    for(CelestialBody &body : system.bodies()) {
        body.velocity += (m_dt/2)*(body.force / body.mass); //Calculate velocity at half step
        body.position += body.velocity*m_dt;
    }
    system.calculateForcesAndEnergy();

    for(CelestialBody &body : system.bodies()) {
        body.velocity += (m_dt/2)*(body.force / body.mass);
    }
}
```

Figure 2: Code for the Verlet integreation solving for position and velocities of each body in the CelestialBody function

2.2 Initialising Code

As mentioned in the introduction, we start with $N = 100$ star particles of masses in a gaussian distribution totaling 10 solar masses, positioned at random within a sphere of radius $R_0 = 20ly$.

Our first task was to initialise the sphere using spherical coordinates and translating this into a cartesian system.

$$x = rsin\theta cos\phi \tag{15}$$
$$y = rsin\theta sin\phi \tag{16}$$
$$z = rcos\theta \tag{17}$$
$$\theta \in [0, \pi], \phi \in [0, 2\pi], r \in [0, R_0] \tag{18}$$

In order to translate these equations back into the Cartesian coordinate system, a set of operations need to be made. We have introduced three new variables with random values between 0 and 1 using a 3x1 matrix. These values will be useful in order to create volume elements in Cartesian coordinates. The translation between these two coordinate systems with these new variables; u, v and w we get that:

$$r^2 sin\theta dr d\theta d\phi = A du dv dw \tag{19}$$

$$r^2 dr = adu, sin\theta d\theta = bdv, d\phi = cdw \tag{20}$$

We find that where $\phi = 2\pi$ and w=1 and the integrated solution is found to be

$$d\phi = cdw \rightarrow \phi = cw \rightarrow c = 2\pi \tag{21}$$

We can now integrate and solve for the other variables from the same approach and the results are as follows:

$$r = R_0 \sqrt[3]{u} \tag{22}$$

$$\theta = arccos(1 - 2v) \tag{23}$$

```
for(int i=0; i< N;i++){
    mat randomPosition = randu(3,1);
    double u = randomPosition(0,0);
    double v = randomPosition(1,0);
    double w = randomPosition(2,0);

    //mat randomVelocity = randu(3,1);
    //double u2 = randomVelocity(0,0);
    //double v2 = randomVelocity(1,0);
    //double w2 = randomVelocity(2,0);

    double phi   = w * 2 * M_PI;
    double theta = acos(1-2*v);
    double r     = R0 * pow(u, 1./3);

    double x = r*sin(theta)*cos(phi);
    double y = r*sin(theta)*sin(phi);
    double z = r*cos(theta);
```

Figure 3: Code for initialising random spherical distribution of star particles.

In the above figure (figure 3) you can see the initialising of these variables and calculating the spherical portions, θ, ϕ and r and inserting them into the equations for x, y and z in order to convert to a Cartesian system. Then, we wanted to initialise the masses of the stars in our system to be within a Gaussian distribution where the mean mass was 10 M_{sun} with 1M_{sun} standard deviation. We did this by using the random number generator and the normal distribution function in QT creator. Once the code was set up and the particles had been generated in random positions with random masses in this distribution, we were able to calculate the forces and energy between these particles.

2.3 Calculating Energy of an Open Cluster

2.3.1 Energy of All Objects

Using the velocity Verlet class to solve for positions and velocities of each object in the cluster, we were able to calculate the forces between each body using the Newtonian gravity equation. In order to calculate the force, we needed to solve for the gravitational constant from the collapse time of the open cluster as seen in equation 9. The following code shows the calculation of the force, kinetic and potential energy of each object. There are two force equations, one with only Newtonian gravity, and one with the implementated smoothing parameter ϵ where;

$$F_{Newt} = -\frac{GM_1M_2}{r^2} \tag{24}$$

$$F_{mod} = -\frac{GM_1M_2}{r^2 + \epsilon^2} \tag{25}$$

```
for(int i=0; i<numberOfBodies(); i++) {
    CelestialBody &body1 = m_bodies[i];

    for(int j=i+1; j<numberOfBodies(); j++) {
        CelestialBody &body2 = m_bodies[j];
        double dx = body1.position[0] - body2.position[0];
        double dy = body1.position[1] - body2.position[1];
        double dz = body1.position[2] - body2.position[2];
        double dr2 = dx*dx + dy*dy + dz*dz;
        double dr = sqrt(dr2);
        double epsilon = 0.05;

        m_G= (M_PI)/((32*numberOfBodies()*avg_mass*t_crunch*t_crunch)/(4*M_PI*R0*R0*R0));

        Vec3 deltaRVector = body1.position - body2.position;
        dr = deltaRVector.length();

        //code for forces on bodies without the smoothing factor
        body1.force += (-m_G*body1.mass*body2.mass*deltaRVector)/pow(dr,3);
        body2.force -= (-m_G*body1.mass*body2.mass*deltaRVector)/pow(dr,3);

        //code for forces on bodies with the smoothing factor
        //body1.force += (-m_G*body1.mass*body2.mass*deltaRVector)/(pow(dr,3)+(epsilon*epsilon));
        //body2.force -= (-m_G*body1.mass*body2.mass*deltaRVector)/(pow(dr,3)+(epsilon*epsilon));

        //m_totalmass += body1.mass+body2.mass;

        m_potentialEnergy += -(m_G*body1.mass*body2.mass)/dr;
    }
    m_totalmass     += body1.mass;
    m_kineticEnergy += 0.5*body1.mass*body1.velocity.lengthSquared();
}
```

Figure 4: Code for calculation of forces and energy of all bodies for N star particles.

2.3.2 Energy of Bound Objects

In our system, many particles are being gravitationally influenced to be shot out into the void. In order to calculate the kinetic and potential energy we wrote a function which removed them from the energy calculations.

```
        // Counting bounded objects
        if (energyiTotal < 0) {
            bodyi.setIsBound(true);
            numberBounded++;
        }
    }
    double totalK = 0;
    double totalV = 0;

    // Compute kinetic and potential energy only of bounded objects
    for(int i=0; i<numberOfBodies(); i++) {
        CelestialBody &bodyi = m_bodies[i];
        if (bodyi.isBound()) {
            // compute kinetic energy of bodyi
            totalK += 0.5*bodyi.mass*bodyi.velocity.lengthSquared();;
            for(int j=i+1; j<numberOfBodies(); j++) {
                CelestialBody &bodyj = m_bodies[j];
                // compute potential(i,j)
                if (bodyj.isBound()) {
                    double dx = bodyi.position[0] - bodyj.position[0];
                    double dy = bodyi.position[1] - bodyj.position[1];
                    double dz = bodyi.position[2] - bodyj.position[2];
                    double dr2 = dx*dx + dy*dy + dz*dz;
                    double dr = sqrt(dr2);
                    m_G= (M_PI)/((32*numberOfBodies()*avg_mass*t_crunch*t_crunch)/(4*M_PI*R0*R0*R0));
                    Vec3 deltaRVector = bodyi.position - bodyj.position;
                    dr = deltaRVector.length();
                    totalV += -(m_G*bodyi.mass*bodyj.mass)/dr;
                }
            }
        }
    }
```

Figure 5: Code for calculation of forces and energy of bound objects in an N star particle system.

When the total energy is negative, then the gravitational binding energy exceeds the kinetic energy and the object is counted as bound. If this condition is satisfied, the code will run for these objects and calculate the total potential and kinetic energy of the system using the same method and G parameter as the code in figure 4.

3 Implementation and Analysis

3.1 Time to reach Equilibrium

As mentioned in section 2.1 the positions can be calculated using the Newtonian gravity equation for force and the velocity Verlet method. We started each particle at rest

and integrated over time in order to produce figure 6, which shows the trajectories of positions of each star particle in the open cluster. We used time steps of 1e-4 in units of τ_{crunch} for a total of 5 τ_{crunch} (40 million years). Larger time steps produce instabilities and smaller time steps increase the computing time unnecessarily. Figure 6 shows the positions of each star particle for a total time of $5\tau_{crunch}$ with distance units in Light Years.

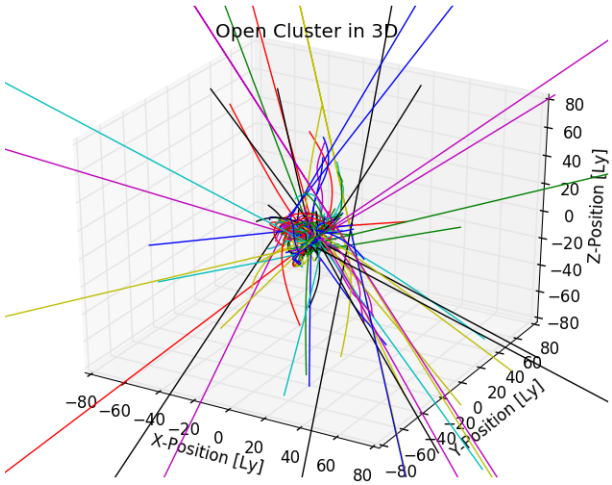
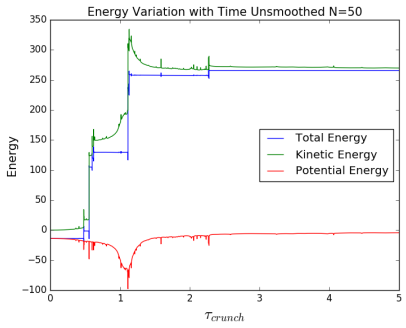
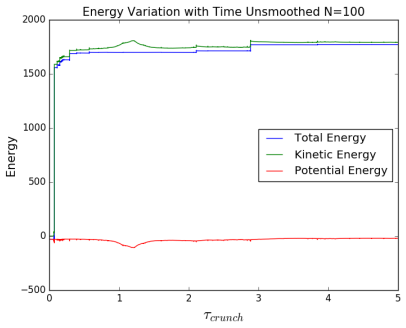


Figure 6: Trajectories of 100 star particles, for $5\tau_{crunch}$ and steps of $10^{-5}\tau_{crunch}$ with distance in Light Years.

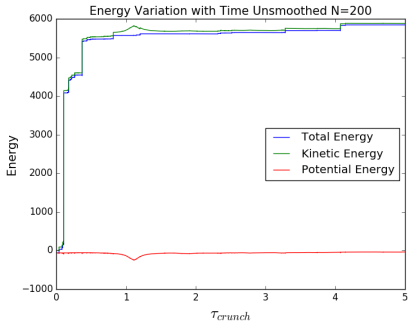
Figure 6 shows that many of the initial star particles belonging to the open cluster are shot out of the system. This occurs when particles come close and interact gravitationally. They then exert strong forces on each other, causing high acceleration and velocities that exceed the escape velocity of the open cluster. We have calculated the velocity that these stars would need to escape, which we found to be V_{esc}= 4275 km/hr. When we look at figure 6, it is very hard to deduce if the system has reached some kind of equilibrium situation, which is what we want to investigate in this section. It is easier to understand what is happening in this scenario by looking at the animation we made, which is called **ClusterCollapseUnsmooth.avi** and can be found in our github repository. In this animation many of the star particles are thrown out of the system, while some appear to be left bound to each other, but again it is not abvious to see if the system is in equilibrium. In order to find out if the system is in equilibrium, we therefore plotted the kinetic, potential and total energies of the system. This can be seen in figure 7, where we have included the energies for systems containing 50, 100 and 200 star particles.



N=50 Stars



N=100 Stars



N=200 Stars

Figure 7: Variation of energies with crunch time for systems of 50, 100 and 200 star particles with no smoothing parameter. The units of energy are $10^4 M_{sun}(ly^2)/\tau^2_{crunch}$ for all energy plots.

As the cluster is collapsing inwards at τ_{crunch}=1 the velocities of the particles increase, and therefore make the kinetic and total energy increase. The potential and kinetic energies peak at τ_{crunch}, which is when the system collapses towards the centre of the gravitational potential well. After this point the system appears to reach an equilibrium at around 2 τ_{crunch} with the exception of slight jumps in total and kinetic energy which represents particles being ejected from the system. Although an equilibrium situation occurs, the energy is not conserved in any of these systems because the total energy varies due to the ejection of particles. Particle ejection causes a significant amount of energy to be lost from the system and is directly dependent on the initial number of particles. This dependency is studied more closely later in the report (section 3.2 and in particular figure 12). In the above figures we used masses of $10\pm1 M_{sun}$ for all star particles, and from the y-axes we see that we get larger differences in energies for larger systems, as more mass is being ejected. We also see that the energy is more "jagged" in smaller systems, as single particles carry more of the total mass and hence affect the total system more strongly.

3.2 Numerical instabilities

In this section we correct for the numerical instability arising when two particles become very close by introducing a smoothing factor epsilon to the Newtonian force of gravity. This makes the force finite at short ranges, so that our forces (and accelerations) do not become unreasonably large. The force equation then becomes:

$$F_{mod} = -\frac{GM_1M_2}{r^2 + \epsilon^2} \tag{26}$$

Where epsilon is the smoothing parameter and has a small value. We experimented with different values of this parameter, as seen in figure 8. We found that a value of $\epsilon = 0.05$ produced the best results, and made a decision to use $\epsilon = 0.05$ for the remainder of the exercises. This was done because this value produced the most energy conserved curve and it is a small value making the modified force close to Newtonian gravity. We later realised that we should maybe rather have used a smaller value, such as $\epsilon = 0.01$, to get rid of the oscillations present after $\approx 2\tau_{crunch}$ and use an equation closer to the Newtonian gravity.

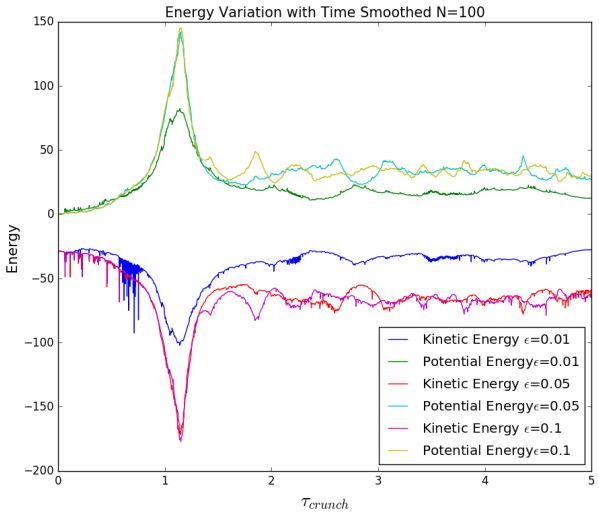


Figure 8: Testing different smoothing parameters for an N=100 system

We can justify adding a correction factor epsilon because our code uses finite time steps, and in real life we obviously would have a continuous interaction. The finite time steps leads to some of the particles being affected by a very strong force for a long time, leading to unphysical accelerations. Real stars also have finite extensions, so that

merging and collisions would happen more frequently in a physical open cluster than in our simulation where the star particles are represented by point masses.

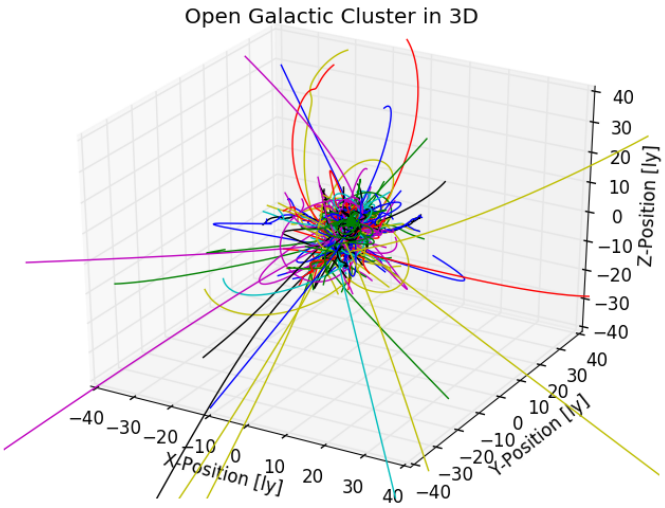
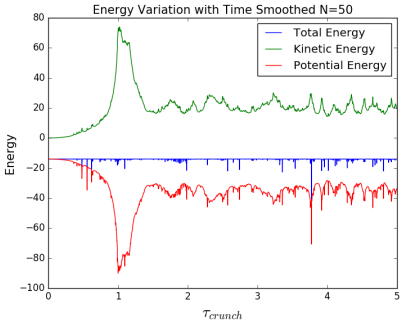


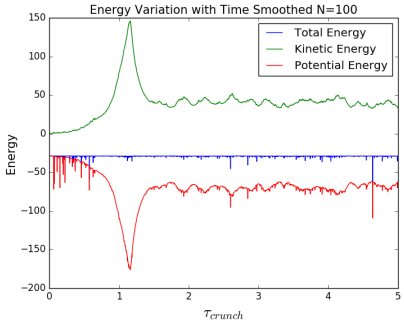
Figure 9: Plots of trajectories of the original 100 particles, for $5\tau_{crunch}$ and steps of $10^{-5}\tau_{crunch}$. Here we have included a smoothing factor of 0.05, and see that slightly fewer particles are lost from the system than in figure 6.

The incorporation of the smoothing parameter changes the results from the previous section. In the above figure (figure 9) we see that slightly fewer particles are lost from the system than in figure 6. Again, we incourage the reader to have a look at an animation of this for better understanding, the animation of figure 9 is called **ClusterCollapse.avi**. We will now also study the energy conservation for systems with different number of particles with the smoothing factor. This was done graphically, in figure 10. In this figure we see that the smoothing parameter has caused conservation of energy and a more defined equilibrium state after τ_{crunch}. The peak in kinetic energy at the collapse time $1\tau_{crunch}$, and the peak in the negative potential energy at the same location represent the particles collapsing to the centre of the gravitational structure. The system stabilises around $2\tau_{crunch}$, and we conclude that equilibrium has been reached. There are however a lot of spikes in the graphs, indicating that energy is not completely conserved and particles are being shot out of the system, as was also seen in figures 6 and 13. Further, we see less numerical instabilities for systems with more particles due to the ejected particles having less of an effect on the system. When initialising our code we chose that the total mass of the system was 1000 M_{sun} which means that if there are fewer particles, then each particle will have a significantly higher mass.

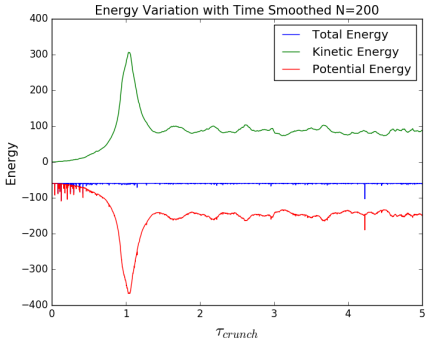
Therefore systems with more particles are less affected by the ejected particles and we see less spikes in the energies.



N=50 Stars



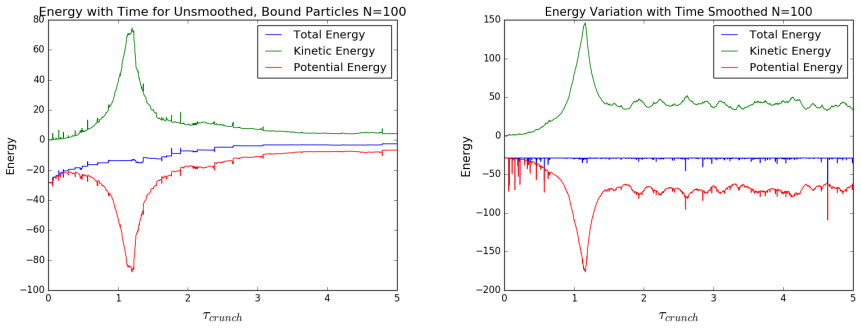
N=100 Stars



N=200 Stars

Figure 10: Variation of energies with crunch time for systems of 50, 100 and 200 particles with a smoothing parameter of $\epsilon = 0.05$.

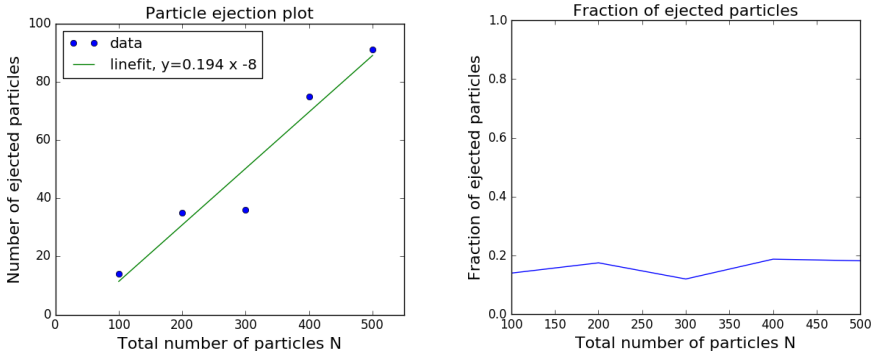
From these figures we see that the smoothing factor does indeed improve our results, a comparison can be seen in figure **??**. It seems that fewer particles are lost from the system, and energy is conserved in the smoothed case. We also see that the total energy of the system is negative, but this is to be expected if the system is in equilibrium and the Virial theorem holds. The validity of the Virial theorem in the open cluster is investigated in the next section.



(a) (b)

Figure 11: A comparison between smoothed and unsmoothed energies for bound particles with N=100. We see that energy is not conserved in the unsmoothed case.

The ejected particles can be identified by their energy; particles on stable orbits will have a total energy of zero (the kinetic energy equal to the negative potential energy), while unstable particles will differ from this. If the gravitational binding energy dominates the particle is accelerating towards the centre, while if kinetic energy dominates the particle is no longer bound to the system and will be shot out. The code for this was shown in section 2.3.2 and figure 5. The number of ejected particles changes with the number of initial particles, as seen in figure 12. Here we see that more particles are lost from larger systems, but we find that the fraction of ejected particles stays roughly constant with N and that on average 16% of the initial particles are lost from the system. As the masses of the particles are roughly the same, $N \propto mass \propto energy$, and we deduce that the fraction of ejected energy is also approximately constant with N.



(a) (b)

Figure 12: In (a) we see a clear relationship between the number of ejected particles and the initial number of particles N. In (b) the fraction of ejected particles is plotted, and we see that on average 16% of the initial particles are ejected from the system. So more energy is lost from larger systems, but the fraction remains the same.

3.3 Test of the Virial theorem

We were able to calculate the kinetic and potential energies of just the bound particles for each system (i.e. particles that have not escaped the open cluster). We investigated these energies for different values of N as seen in figure 13.

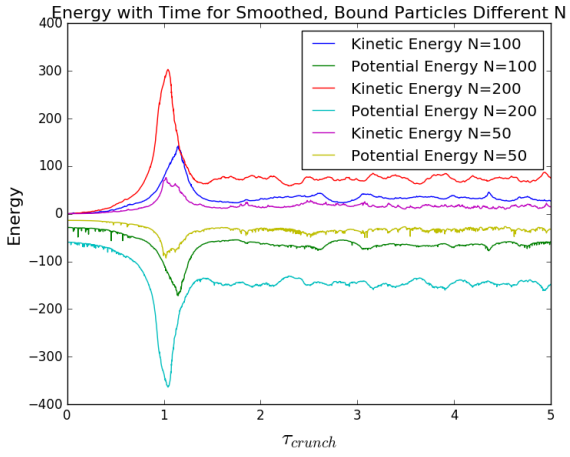


Figure 13: Potential and Kinetic Energy for different number of particles in bound systems with smoothing parameter of $\epsilon = 0.05$.

We have studied the validity of the Virial theorem for the particles that have not been ejected is to further understand the state of the open cluster. The Virial theorem states that for a gravitational system in equilibrium, the absolute value of the average potential energy is equal to two times the average kinetic energy:

$$2 < K >= - < V > \tag{27}$$

To see if the Virial theorem is valid and our system is in equilibrium, we plot the left hand side and the right hand side of this equation, see figure 14.

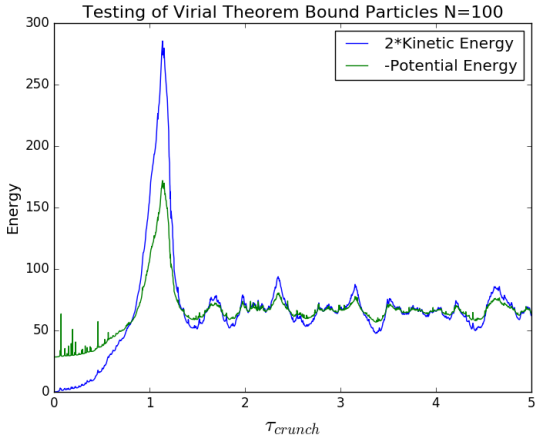
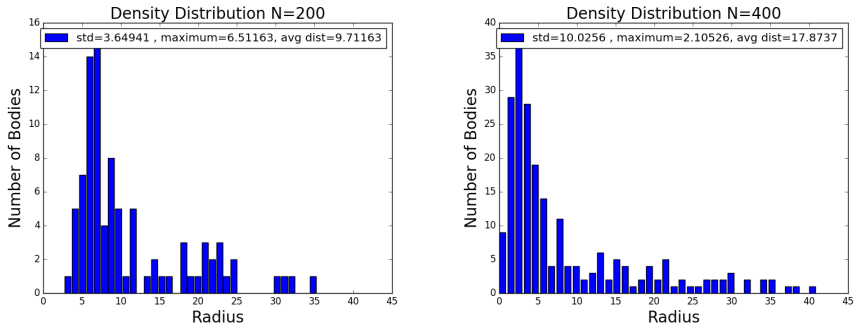


Figure 14: Testing the Virial theorem by comparing the left and right hand side of equation (27), two times the kinetic energy and minus the potential energy.

We can see from this figure that before $\tau_{crunch} = 1$ the potential energy is larger than the kinetic energy, meaning gravity is winning and the system is collapsing. During the collapse at approximately $\tau_{crunch} = 1$, we see a significantly larger kinetic energy than potential. This is due to the high velocity of particles as they are collapsing to the center of the well. Though once we reach $\tau_{crunch} = 2$ the two sides of equation (27) match, which proves we are in an equilibrium situation and the Virial theorem is satisfied.

3.4 Density of particles

We now investigate the radial density of particles and it's dependency on the number of particles in the system. We first make histograms of the density distributions of particles for systems of 200 and 400 initial particles, as seen in figure 15. This shows that when particle mass is kept constant, the average radius of the system at equilibrium decreases with N, this can be explained by the larger mass and hence also binding energy of the system.



(a) N=200, $R_{maxparticles}$=6.51, σ=3.65 (b) N=400, $R_{maxparticles}$=2.11, σ=10.03

Figure 15: Histograms showing the radial distributions of particles in light years after $5\tau_{crunch}$ for systems of initial number of particles of (a) 200 and (b) 400. Here the masses of all particles were 10±1 solar masses, so the total mass of the cluster was larger for larger N. We see that the particles are distributed closer to the centre of the sphere with a smaller standard deviation for the N=400 case than for the N=200 case. This can be explained by the larger gravitational force of the cluster.

In figure 16, we see that the radial position increases with number of particles in the simulation. In these plots the total mass of the system has been kept constant rather than the particle mass. This is interesting to note, and means that the more particles we simulate to represent for example a dark matter halo of a given mass, the larger the radius of the virialised halo. Ideally we would like some way of simulating these conditions that is not so dependent on the number of particles. This is because it is rather hard to count the dark matter particles in over densities in the early universe, as we do not know what these particles are, we can not see them, and we do not know the conditions in the early universe very well. In this figure it should be noted that we plotted quite few points, so the data are not completely reliable, but suggest a linear relationship. The relation between the radius where the most particles reside and the number of initial particles is investigated in figure 17. This shows that the line fits reasonably well, but that more data are needed to be certain about this relation.

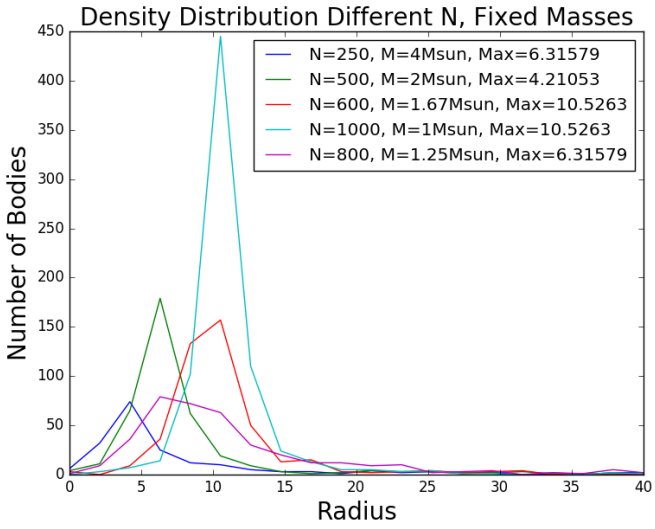


Figure 16: Distribution of number densities of particles for N=250, 500, 600, 800 and 1000 particles. The total mass of the system is kept constant, so the mass of the star particles is smaller with larger N.

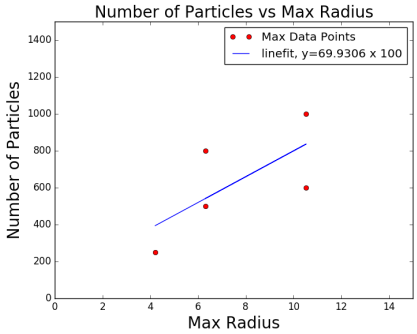
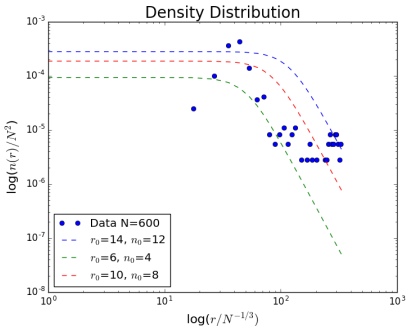


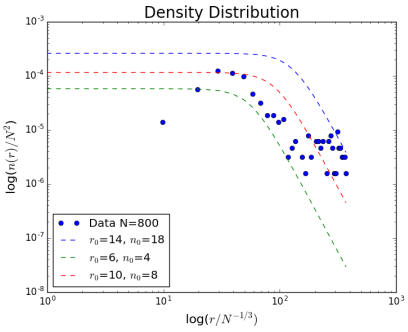
Figure 17: Linefit to the maximum datapoints in figure 16.

We then wanted to fit our number density distribution with the following function. This was also done by [5]. Our results can be seen in figure 18.

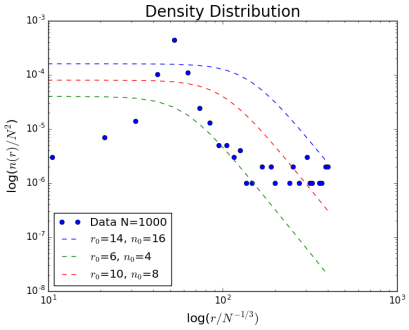
$$n(r) = \frac{n_0}{(1 + (\frac{r}{r_0})^4)} \tag{28}$$



(a) N=600



(b) N=800



(c) N=1000

Figure 18: Number density distribution of particles with radius for N=600 (a), N=800 (b) and N=1000 (c). We see that we get a reasonable fit with equation (28), The values for r_0 and n_0 were found from trial and error with the density profile function. The axes are however scaled with N^2 (y) and $N^{-1/3}$ (x), so we can deduce that the parameters scale with N accordingly. The density function was scaled with N^2 (y) and $N^{-1/3}$ (x) when plotting.

From these figures it is hard to deduce a clear dependency of r_0 and n_0 on N, but the axes are scaled with N^2 (y) and $N^{-1/3}$ (x), as these were the results found by [5]. As the fit does not seem to be entirely bad, we can also argue that the parameters scale with these values. The reason it is so hard to see a dependency from our data is most likely that our maximum number of particles was only 1000, while [5] used many hundred thousand particles. We conclude from this that a higher number of particles are required to find a clear relation.

4 Conclusion

In conclusion, we have found that open clusters such as NGC 265 will not collapse into a point mass with time, as is theoretically predicted to be the case for a completely uniform spherical distribution of particles ($N \to \infty$). Instead we see that a significant fraction of the stars are thrown out of the system, before the system stabilises at a finite radius. This Virialisation happens around $2\tau_{crunch}$. The system then reaches a steady equilibrium situation where the gravitational pull of the stars is balanced by their velocities at a Virial radius. When the system is in equilibrium particles are still occasionally shot out of the system if they come too close to each other, and this is most likely the reason of the short lifetime of open galactic clusters.

The Virial radius, as well as the fraction of particles that are thrown out of the system, depends strongly on the initial number of particles N. We find that the radius of the virialised structure scales linearly with N when total mass is kept constant, and that the fraction of ejected particles is roughly constant at 16%. We also find a good agreement between our density profile and equation 28, and that the parameters in this equation scales as $r_0 \propto N^{-1/3}$ and $n_0 \propto N^2$ as was also found by [5]. These results might not be valid for very large N, we only simulated 1000 particles and [5] did a few hundred thousand, but if this is true it means that we cannot simulate continuous fluids just by increasing N. We would then have to turn to some other method than N-body simulations, or insert some other kind of interaction between the particles.

By completing this project we have learnt that N-body systems with a finite number of particles do not collapse into a point in a time τ_{crunch} as is expected for a continuous fluid, and that understanding such systems could be key to understand the early universe and many other important fields in Physics and Science in general. Simulating continuous fluids is not trivial, and more advanced programming techniques are required to be able to do this successfully.

In future work, a better algorithm should be used in order to include more particles. Parallelization may also be considered to reduce computing time. Numerical methods that might be applied to increase number of particles and reduce computation time could be for example the Barnes-Hut tree method and/or the particle mesh method. These were utilised in the now famous Millennium run simulation of the Universe [2] which used a version of the GADGET 2 code from [6] to run a simulation with 10 billion particles ($N = 10^{10}$) to look at large scale structure formation in the early universe.

References

[1] Solar System Project. https://github.com/akspilke/PROJECT3-Solar-System. Accessed: 2016-12-04.

[2] Michael Boylan-Kolchin, Volker Springel, Simon DM White, Adrian Jenkins, and Gerard Lemson. Resolving cosmic structure formation with the millennium-ii simulation. *Monthly Notices of the Royal Astronomical Society*, 398(3):1150–1164, 2009.

[3] Mark Dijkstra. Lecture notes in cosmology and extragalactic astrophysics.

[4] HST. NGC 265. http://spacetelescope.org/images/heic0603b/. [Online; accessed 30-Nov-2016].

[5] M Joyce, B Marcos, and F Sylos Labini. Cold uniform spherical collapse revisited. *arXiv preprint arXiv:1011.0614*, 2010.

[6] Volker Springel. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005.