N-body Simulations at the Globular Cluster scale

Matthew Rayner

December 21, 2018

Abstract

The mechanics of many-body systems (primarily globular clusters) are explored in this project. Clusters were generated using Monte-Carlo rejection techniques, then simulated using a Hermite-integration N-body program. A variety of starting conditions were used to create different systems, with analysis on both position and velocity-space. The data are presented as examples of Cluster evolution (and collision) in the mega-year (Myr) scale, with discussion on various phenomena including particle escape, evaporation, core collapse and cluster relaxation.

Contents

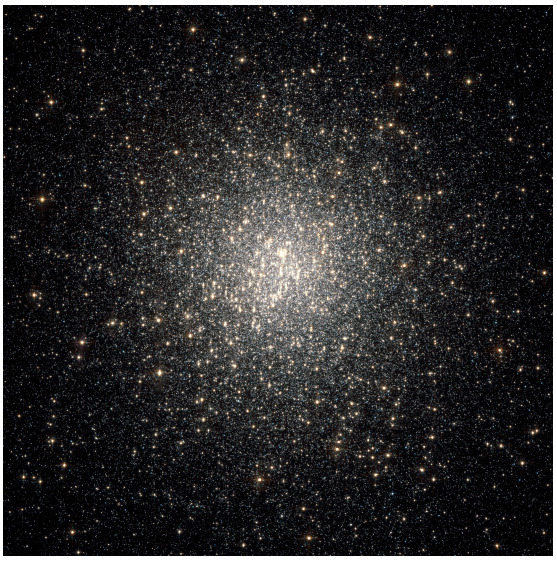


Figure 1: Globular Cluster NGC2808 [1]

1 Introduction

N-body simulations are used to explore the effects of gravity between systems of many stellar objects (referred to as particles throughout). This is relevant at a variety of scales, including solar systems (and exoplanetary systems), stellar clusters and galaxies. All of these systems are 'non-collisional', meaning that the probability of each body colliding is so close to 0 that it is approximated as 0; particles are treated as points.

The primary focus of this project is at the globular cluster scale. Globular clusters are gravitationally bound systems of about 10 thousand to 1 million stars (figure 1 is an example of the latter), spread over a diameter of around 25 to 200 light-years [2]. As such, they exist between the scales of the solar system and galaxy. There are roughly 150 known globular clusters in the milky way [3], with several more potentially undiscovered. The sizes and shapes of these clusters can vary greatly, and some probably even contain black holes [4]. This gives strength to the theory that super-massive black holes at the centre of galaxies are formed from the merger of smaller black holes, held by globular clusters (such as the cluster in galaxy M31 from this study).

In Toomre and Toomre's landmark paper "Galactic Bridges and Tails" [5], galactic-scale simulations were run to explore the effects of galactic collision. The authors argue that some of the 'tails' and 'bridges' observed in various galaxies are remnants of close encounters between galaxies. The paper used FORTRAN code to simulate the galaxies, which are approximated as rings of particles around a large central mass.

46 years after "Galactic Bridges and Tails", computing power has increased nearly exponentially, as predicted by 'Moore's Law' [6]. Due to this, computationally heavy simulations have become increasingly popular and accessible; Most of the work presented in this report has been done on a laptop and desktop machine that would have been classified as a supercomputer in the 70s, and in a fraction of the time. The codes used in this report are in C or C++, languages that were partially influenced by FORTRAN, but arguably much more efficient.

In this report, globular clusters are presented using the Plummer Model [7], and (inspired by Toomre & Toomre) collided together. Various phenomena are observed during these collisions. The method of the report is written as a guide on the entire process, so that the reader can attempt to follow it and create their own simulations.

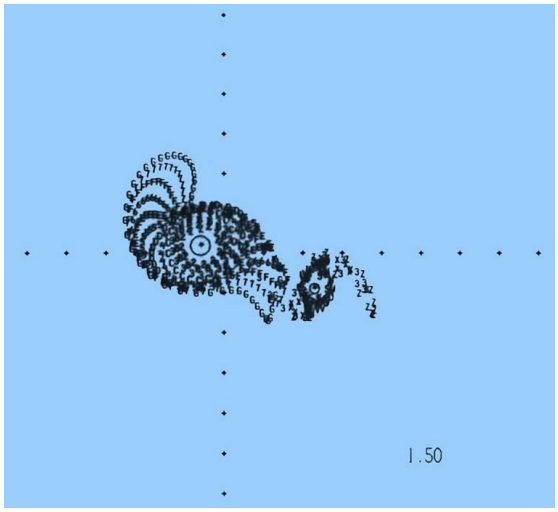


Figure 2: Still from Toomre & Toomre's movie [8]

2 Theory

2.1 The Plummer Model - position distribution

Plummer proposed that stellar clusters followed a specific potential (the "Plummer Potential") of the form:

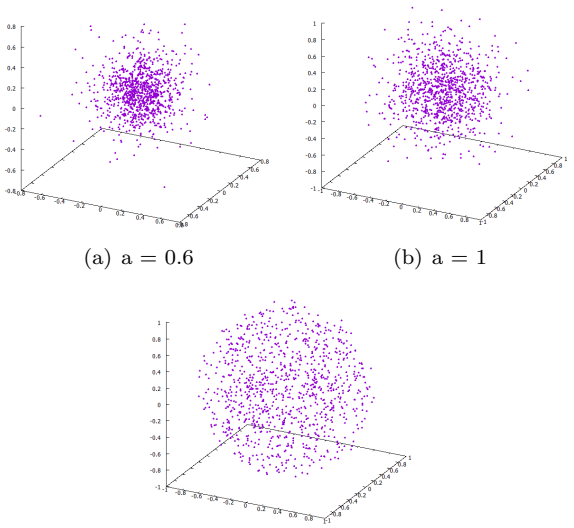
$$\psi(r) = -\frac{GM}{\sqrt{r^2 + a^2}} \tag{1}$$

Where G is the gravitational constant, M is the total mass of the cluster, r is the distance of a specific particle from the centre of mass, and a is a constant that determines the size/concentration of the cluster: Smaller values of a increase the central density of the cluster.

For this potential, the corresponding density distribution is:

$$\rho(r) = \left(\frac{3M}{4\pi a^3}\right)\left(1 + \frac{r^2}{a^2}\right)^{-5/2} \tag{2}$$

(See **Appendix A** for a full derivation)



(a) a = 0.6 (b) a = 1

(c) a = 5

Figure 3: Plummer Model (1000 points) with different 'a' values

This model is created with a few approximations that are held throughout the project: $M = 1$, so that each particle has mass $1/N$ (where N is of course the number of particles being generated). 3(a) is the most tightly bound of the 3 examples here. Once a reaches a value over around 3, the cluster looks like a uniform distribution (as in 3(c)). The maximum value that r can take is also 1 (in code units).

2.2 The Plummer Model - velocity distribution

Whilst it would be easier to simply calculate a position (density) distribution for stars in a cluster and set their velocities to 0, this would completely ignore the fact that these particles contain a mixture of potential AND kinetic energy. Instead, it is possible to populate each particle's velocity according to:

$$v = c * \sqrt{2} * (1 + r^2)^{-1/4} \tag{3}$$

where c is a randomly sampled number between 0 and 1, that also satisfies the condition:

$$c^2 * (1 - c^2)^{7/2} < d \tag{4}$$

where d is a randomly sampled number between 0 and 0.1 (see [13] for a full derivation). In the results section, simulations will be presented as either "hot" or "cold". This refers to whether or not the particles have been assigned velocities (hot) or not (cold).

2.3 Relaxation Time

In an astrophysical context, the relaxation time of a system of stars is "the time taken for a star to lose all memory of its initial orbit" [10]. In this report, a proposed proxy for relaxation time is presented: that of the velocity space average radius for a cluster. The velocity space is a visualisation of the velocity values for each point in a cluster, and can fortunately be extracted from the simulation data very easily. Once the radius of velocity space becomes (close to) constant, the cluster has reached a relaxed state.

3 Method

The process of creating visualisations of simulated globular clusters in this report is split into 4 main steps:

1. Generating clusters and initial conditions.
2. Using "GravityLab" (nbody_sh1.c) to set the clusters in motion, and generate data for processing.
3. Processing data into a single set to show positions at specific time points,
   OR splitting data into a series of sets to make an animation (.gif).
4. Using gnuplot, with the processed data, to visualise the clusters.

3.1 Step 1) Generating clusters

Globular clusters were generated using a 'guided' random generation, with properties described in the Theory section. See **Appendix B** for a full code listing. The program generates a random value of r between 0 and 1, then calculates the cube root of this value (to ensure uniformity in 3 dimensions). This value is then used to calculate the Plummer Density

Letter	No. of Clusters	N (per cluster)	a	Sim time	hot/cold?	x shift	v_x shift	v_y shift
a)	1	100	1	15	hot	-	-	-
b)	1	80	1	12	cold	-	-	-
c)	1	100	5	15	hot	-	-	-
d)	1	100	5	11	cold	-	-	-
e)	1	80	0.6	9	hot	-	-	-
f)	2	100	1	15	hot	2	0.25	0
g)	2	100	1	15	hot	2	0.2	0.2

Table 1: Simulation List

$\rho(r)$. If $\rho(r)$ is less than *another* randomly generated value (in the same manner as r), then the point is rejected, and the loop is restarted.

If the point *is* accepted, then its velocity (magnitude) is calculated in the v_calc() function, according to the requirements presented in the Theory (section 2.2). Both functions are examples of simple Monte Carlo rejection techniques (see [9] for in-depth theory behind this method).

Once these values are generated, they are converted into Cartesian form (by randomly assigning each point a ϕ and θ angle first). The values are then put into a file of the user's choice, in a format ready for Step 2).

To use the program, first compile it (I personally used MinGW for windows), and run. The program first asks the user what to name the output file, then asks how many particles to generate, followed by a value of a.

3.2 Step 2) nbody_sh1.C

The file generated by sph_gencluster.c is of the form:

```
N
t
m1  x1  y1  z1  vx1 vy1 vz1
m2  x2  y2  z2  vx2 vy2 vz2
...
mN  xN  yN  zN  vxN vyN vzN
```

Where N is the total particles, t is the start time, $m1$ is the mass of the 1^{st} particle, $x1$ its x position, $vx1$ its x velocity etc. This is the expected input for "nbody_sh1.C". (Note: this is a C++ program and should be compiled as such).

For almost all simulations done in this project, nbody_sh1.C was used in the following way (using a console):

```
nbody_sh1 -d 0.04 -o 0.04 -t 15 < input.in
    > output.out
```

This means the program will calculate and print output (to a created file named "output.out") 25 times per time-unit, for 15 time units using the input file "input.in". For a more detailed guide on using "nbody_sh1.C", refer to [11] and the code itself [12], which gives a reasonable explanation for its use.

3.3 Step 3) Data processing

After running "nbody_sh1.C", an output file is generated, with the same format as the input, but much larger as it contains information for each time-step. At this point it was useful to process the data into its x, y, and z components (or even its velocity components to look at the clusters velocity-space). This was done through the use of awk:

```
awk '{print $2 " $3 " $4}' output.out
    > output.plot
```

for UNIX terminals, or:

```
awk "{print $2 """ $3 """ $4}" output.out
    > output.plot
```

for windows (cmd) terminals. The 2nd, 3rd and 4th columns contain x, y, and z values. The 5th, 6th and 7th columns are used for velocity values instead.

Plotting (using for example GNUplot) at this step will show the particle's orbits, or velocity 'orbits' (for lack of a better phrase). To better look at a system's progression through time, or at a specific time value, a simple 'splitting' program was created called 'splitter2.c' (see **Appendix C** for a full code listing and explanation of use). Using this program results in a series of 'snapshots' for each time step, of the form 'snap_0.txt', 'snap_1.txt' etc.

3.4 Step 4) Data visualisation

After processing the data as required, GNUplot was then utilised to visualise the data.

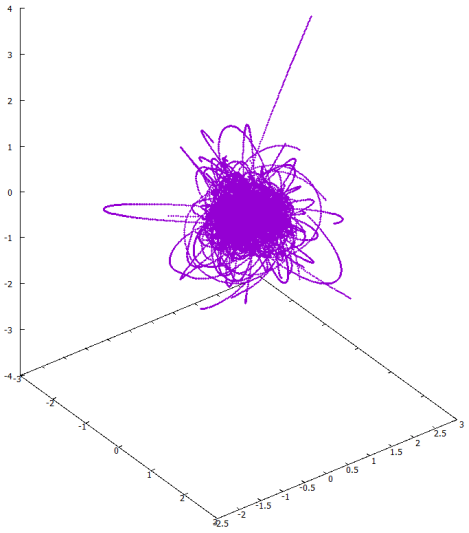


Figure 4: Full orbit visualisation for 100 particle sphere with a = 1

3.4.1 Orbit visualisation

Omitting the 'splitter' option in Step 3) and immediately plotting the output '.plot' file results in orbital graphs that show each time-state of the particle in one plot. If this is desired:

```
(>gnuplot) set view equal xyz
(>gnuplot) set ticslevel 0
(>gnuplot) splot "output.plot" pt 7 ps 0.2
    notitle
```

Will produce a scale-equal graph. See Figure 4 for an example (with axis limits redefined).

3.4.2 Snapshot visualisation

The same process as above was used to create a snapshot image of a desired time, except replacing "output.plot" with "snap_.txt".

3.4.3 Animation

To create a .gif animation using all available snapshot files, a couple of additional steps were required: Firstly, a simple .awk setup script was used (see **Appendix D**). Note that this script is not infallible, and should be changed according to the level of zoom, or amount of clusters in the simulation. Secondly, GNUplot was given a series of files (snapshots) to print one after another into the .gif animation:

```
(>gnuplot) load "gif_setup.awk"
(>gnuplot) do for [i = 0:374]
    {splot sprintf("snap_%d.txt", i)
    pt 7 ps 1 notitle}
```

If done correctly, this produced a .gif with 374 frames that clearly showed the clusters progression through time.

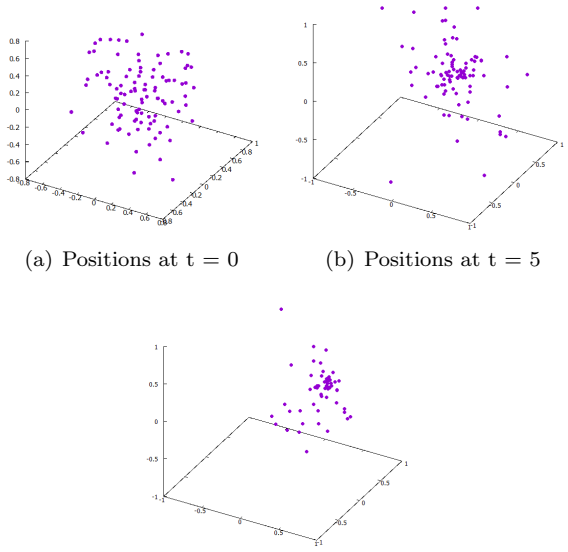
4 Results

Results are presented in pictorial form at different time points. The author also invites the reader to follow the url provided to view .gif versions if interested (See **Appendix E**, if a picture is worth a thousand words then a video is worth a thousand pictures!). See Table 1 for a list of experiments.

In order to cut down on simulation time, a maximum of 100 particles were used per cluster. Interestingly, the value of a was inversely proportional with time taken per simulation; as a decreased, simulation time increased. This is why for $a = 0.6$, only 80 particles were used. Simulation time was also (much) longer for cold start systems. If a simulation was taking over 6 hours to complete, it was ended at the point it had got to. This was the reason some simulations had maximum t values less than 15.

Velocity space is also used to illustrate the behaviour of each cluster: Note that in general, relaxation times for clusters vary based on their a values, and whether they are initially hot or cold. The average radius of the velocity-space sphere is a reasonable indicator for how much kinetic energy the cluster has. Sometimes this 'velocity sphere's' radius increases (during cluster collapse), or settles (during periods of relaxation). Although it is not possible to represent in images alone: once a point escapes a cluster with a certain velocity and retains that velocity, it is represented by a stationary point in velocity space. In most animations, there are a few points that behave like this, and they overlay the more 'noisy' velocity points that are still part of the cluster.

4.1 Simulation a): 100 Particles, a=1, hot.



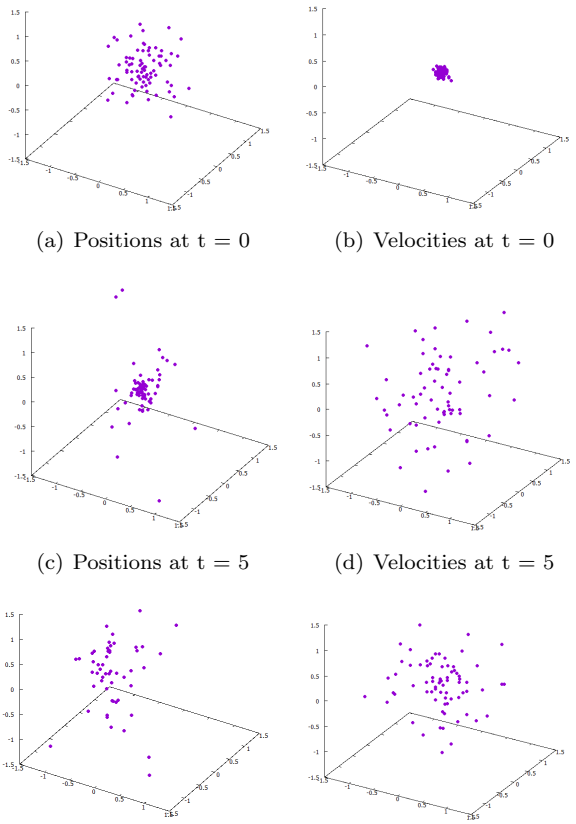
(a) Positions at t = 0 (b) Positions at t = 5

(c) Positions at t = 15

Figure 5: a) 100 Particles, a = 1, hot.

Velocity space is omitted for this section, as it remains relatively consistent throughout after an initial 'adjustment' period where it doubles in size. This means that the initial velocities weren't quite representative of a relaxed system. The position of the centre of mass of the cluster moves quite far throughout the simulation, indicating that some higher-level condition check (like 0 avg. net velocity or 0 avg. centre of mass) has not been checked when the cluster was generated.

The cluster is not immediately relaxed. By t = 5 in 5(b) the cluster has a much more steep density profile than before; the core has collapsed into a smaller area.



(a) Positions at t = 0 (b) Velocities at t = 0

(c) Positions at t = 5 (d) Velocities at t = 5

(e) Positions at t = 11 (f) Velocities at t = 11

Figure 6: b) 80 Particles, a = 1, cold.

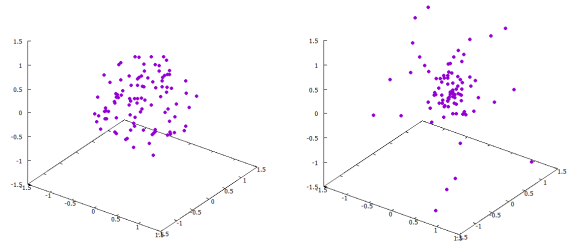
Many points have reached escape velocity by the end of the simulation, reducing the mass of the gravitationally bound area to the point that by $t = 11$, the system literally can't hold itself together (like the author after attempting 2 all-nighters in a row). In 6(e), there is no tightly bound core to the cluster, and given more simulation time it would be expected that the cluster would evaporate entirely.

4.2 Simulation b): 80 Particles, a=1, cold.

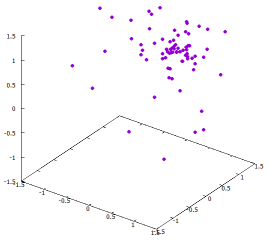
This system should be an example of how globular clusters *don't* behave in reality. Although the plummer potential is followed for position, velocity is completely disregarded. One frame after the start of the simulation, the velocities of all the particles are almost 0, as seen in 6(b). All points rapidly accelerate towards the centre of mass of the system, and in doing so the velocity space starts to rapidly expand, as shown in 6(d). This is the system attempting to put itself into equilibrium, but it is an overcompensation.

4.3 Simulation c) 100 Particles, a=5, hot.

For $a = 5$, the cluster is almost entirely uniform and homogeneous at the start. Surprisingly, the velocity-space for this system is even more stable than in simulation a), and is therefore omitted as it is unremarkable. This was unexpected, as you can clearly see the cluster collapsing quite dramatically into a tight, relatively stable core. With this, it would seem only logical that a change in average velocities would occur. However, if you look at the animation of this system, it is by far the 'calmest', and the evolution



(a) Positions at t = 0 (b) Positions at t = 5



(c) Positions at t = 15

Figure 7: 100 Particles, a = 5, hot.

happens without (for want of a better word) violence.

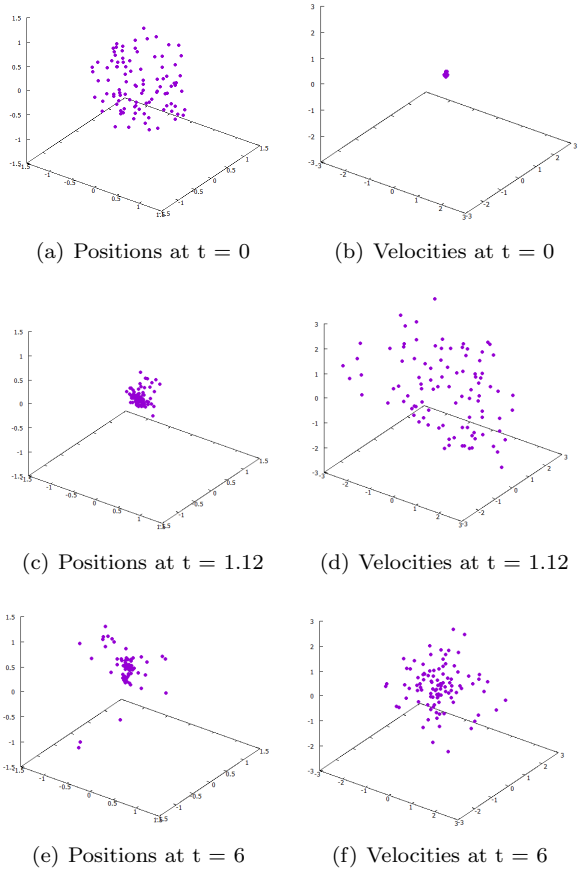
The 'tail'-like structures seen at $t = 5$ are not particles escaping the system. In fact, this was simply a coincidence. Several particles in relatively stable distant, tangential orbits happened to align in these patterns. Once again, the entire cluster has a non-0 avg. velocity, so is displaced from (0,0,0) by the end of the simulation.

4.4 Simulation d) 100 Particles, a=5, cold.

In stark contrast to the graceful behaviour of it's hot counterpart, this system exhibits the most extreme change in velocities, and the most prominent core collapse of all the simulations. It is presented as another example of what too look out for as an *inaccurate* globular cluster simulation: such spikes in velocity like this, as seen in 8(d) are simply not observed in reality.

Many particles are ejected during the initial collapse, and the resultant dense core causes unpredictable ejection of particles afterwards. Also note that as compared with previous velocity-space graphs, these are zoomed out twice as far. The axes span from -3 to +3, as opposed to -1.5 to +1.5. This makes 8(d) seem even more unstable, as velocities are spread out across the entire volume of the graph.

The system does eventually become relatively stable, albeit with a dense core made of fast particles. Therefore a snapshot after $t = 6$ has been omitted. The system also has a net drift velocity, which is not unexpected at this point, as non-0 drift velocities have been a trend in all simulations thus far.



(a) Positions at t = 0 (b) Velocities at t = 0

(c) Positions at t = 1.12 (d) Velocities at t = 1.12

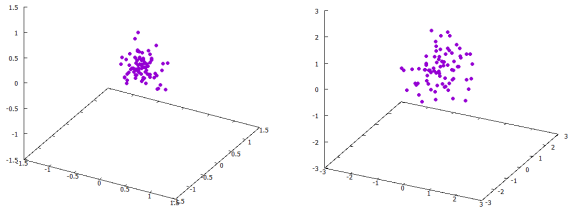
(e) Positions at t = 6 (f) Velocities at t = 6

Figure 8: d) 100 Particles, a=5, cold.

4.5 Simulation e) 80 Particles, a=0.6, hot.

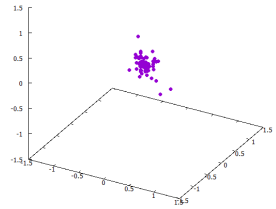
A value of $a = 0.6$ is more realistic for a globular cluster than $a = 1$, and especially $a = 5$. This system starts more tightly bound than the other examples. For 100 particles, the odds of a point being populated outside of -0.5 to +0.5 are very small. So small that in this system, almost all particles are within these bounds to start. The cluster experiences a very fast collapse, as shown by 9(c) and 9(d), where positions converge, whilst velocities disperse. Note that the velocity axes bounds are also from -3 to +3 in these results.

After this, the cluster becomes relatively stable, and begins to drift. However, by $t = 9$, the cluster is

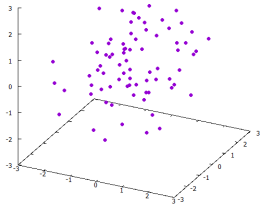


(a) Positions at t = 0

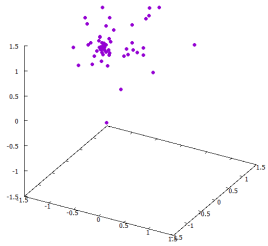
(b) Velocities at t = 0



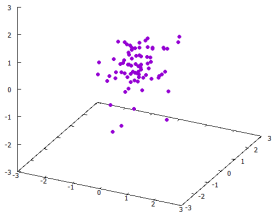
(c) Positions at t = 0.4



(d) Velocities at t = 0.4

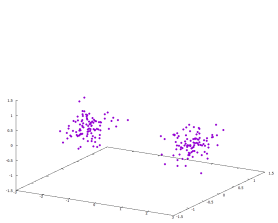


(e) Positions at t = 9

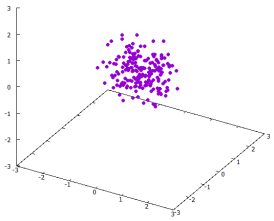


(f) Velocities at t = 9

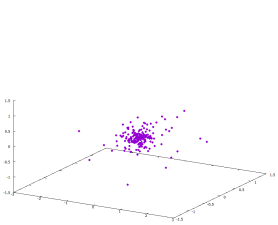
Figure 9: 80 Particles, a=0.6, hot.



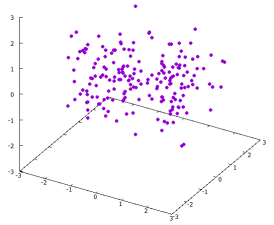
(a) Positions at t = 0



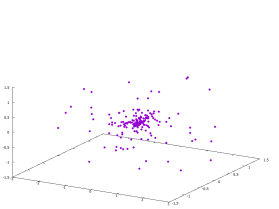
(b) Velocities at t = 0



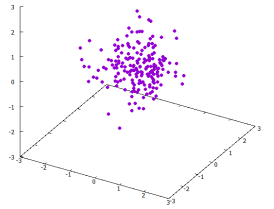
(c) Positions at t = 3.72



(d) Velocities at t = 3.72



(e) Positions at t = 6



(f) Velocities at t = 6

Figure 10: f) 100 Particles per cluster, a=1, hot, 'head on' collision.

experiencing mild evaporation, as well as some particles moving into more distant orbits. In the animation, you can clearly see some points being violently ejected towards the end. The velocity space appears to be stable at this point however, indicating a (relatively) relaxed system.

4.6 Simulation f) 100 Particles per cluster, a=1, hot, 'head on' collision.

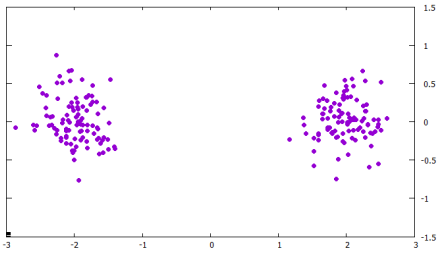
This was the first of 2 simulations with cluster collisions. In this example, both clusters were given an additional x-velocity of 0.25 towards each other. Without gravitational interactions, these two would have collided at about $t = 4$, but they start to accelerate towards each other, hence why 10(c) occurs slightly before $t = 4$. The velocity-space graph [10(d)] at this point isn't exactly spherical, but somewhat like a fuzzy horizontal hourglass. It indicates that each sphere is experiencing a pull from the other.

After the collision at $t = 3.72$, many points are 'shed' from both systems. Although the image is small here, it is possible to make out dots spread across the entire volume of the graph. Due to the extreme acceleration of most particles at the collision point, many are sling-shot outwards.

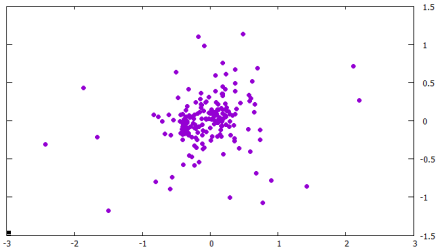
After the collision, a large central cluster is formed, and the system relaxes to a certain degree, as indicated by 10(f). From around $t = 6$ to 15, the system is relatively stable, hence the omission of the end state of the system.

4.7 Simulation g) 100 Particles per cluster, a=1, hot, 'spiraling' collision.

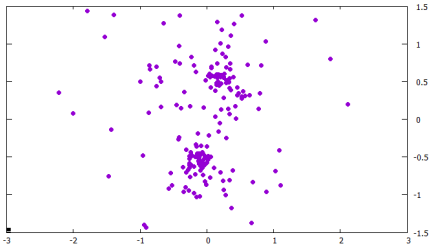
This simulation exhibited the most interesting behaviour. The cluster cores perform a complex dance together, spiralling towards a close approach before retreating for a moment and finally uniting in a spectacular display. Both clusters sacrifice (eject) many particles in the process. It is easier to do the simulation justice by watching the animation, so if there



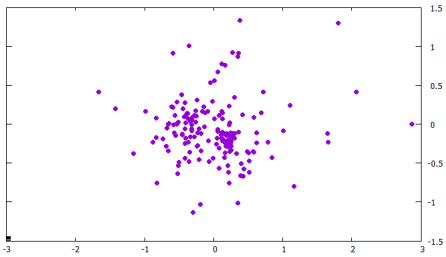
(a) Positions at t = 0



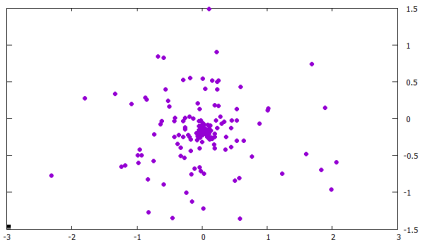
(b) Positions at t = 4.52



(c) Positions at t = 6



(d) Positions at t = 8.4



(e) Positions at t = 15

Figure 11: g) 100 Particles per cluster, a=1, hot, 'spiral' collision.

was only one to watch, it would be this!

The isotropic view presented with the other clusters obscures much of the intricacy. Instead, a top-down view of the x-y plane is used for the position displays. Velocity space is also omitted for this section for the sake of space (but can be found in the animation link).

The clusters are initialised to run anti-parallel to each other [11(a)]. However, they are close enough to experience a significant gravitational pull. After rotating clockwise around each other for roughly 4.5 time units, they come 'face to face' in 11(b), at which point they are sling-shot away from one another to a large distance, in 11(c). In the process, several particles are ejected due to extreme tidal forces.

They are then accelerated towards each other once again, and after around $t = 8.4$, they merge. In the process, once again, particles are ejected or sent into large, elliptical orbits. By the end of the simulation, the large merged cluster behaves as a single entity, and has even undergone relaxation into a stable system.

5 Futher Discussion

There are various considerations, changes, and potential extensions to acknowledge here.

5.1 Error and Uncertainty

The "nbody_sh1.C" program sends a diagnostic report to the screen at a default rate of once per time unit. This diagnostic includes a notification of kinetic, potential and total energy of the system, as well as an approximate error in total energy at each point in the simulation. These values should have been converted to percentages, and included in the report. However, doing this would have meant re-simulating the exact same scenarios again and recording the values, which would have taken a significant period of time...

5.2 Relative and Absolute units

The entire report has used relative units throughout. Unfortunately the absolute units of each simulation change based on the mass, and number of particles in each system. The program also assumes the gravitational constant G to be 1, which is concerning. A possible amendment to the report would be to work out a way of converting simulation units to real units.

This would be necessary to gain a better understanding of the time-frame for these simulations, and the author apologises for the omission of this important step.

5.3 Lack of detailed comparison to observables

Due to the time-scales and nature of this work, it is difficult to check whether these simulations progress accurately through time. The actual progression of individual systems happens on an unfortunately large time scale, meaning that comparison is fundamentally limited. However, it may be possible to do something similar to [14], where a simulation is run and cross-referenced to various real-life observables at each stage. There is no reason why this can't be done at a globular cluster scale as well.

5.4 Single-threaded simulation

All simulations in this report were done using a program that runs on a single thread of a CPU. Without straying too far into computer science territory, this means that in an 8-core CPU (for example), less than 15% of it is being used to compute. This has drastically increased simulation time. The author attempted to optimize the program for use on multi-cored CPUs (i.e. almost all modern ones), but found that this was far beyond his ability. Therefore a proposed extension could be to find a way to increase the efficiency of the program. This would allow higher N simulations at a faster rate.

5.5 Clusters with black holes

As mentioned in the introduction, it is possible that some globular clusters contain black holes. A possible extension for this experiment would be to run scenarios with a much larger central mass, and see if there are any observables to reference with these simulations.

5.6 Accounting for dark matter

Another possible extension could be to account for dark matter and compare simulations with/without this consideration.

6 Conclusions

The nature of this project meant that it was difficult to follow traditional experimental method and structure: Instead of setting up a physical system and directly measuring physical results, a digital system was set up with inferred, virtual results. The codes written and presented in the Appendix, along with the Method, should allow a student at a similar level to quickly repeat the work in this project for themselves, and extend it if desired.

Without being able to directly compare experiment to reality the report cannot present pure, quantitative results. It has, however, presented a model for running moderately complex and initially accurate large scale simulations with suggestions for further extension. It has also provided a detailed description of the dynamics of globular clusters, by making use of velocity-space. There is room to further explore velocity as a tool for measuring characteristics and properties of these clusters. Various different clusters have been explored, and a detailed look into the collision of such systems has been presented.

The author hopes that the data and results have given insight into a scale that we don't usually have the privilege of seeing or studying.

References

[1] NASA, ESA, A. Sarajedini, G. Piotto, (Released May 2007): *Globular Cluster NGC2808* http://hubblesite.org/image/2124/news_release/2007-18

[2] Frommert, H., Kronberg, C., (July 2014): *Globular Star Clusters* http://www.messier.seds.org/glob.html

[3] Frommert, H., (August 2007): *Milky Way Globular Clusters* http://spider.seds.org/spider/MWGC/mwgc.html

[4] Finley, D., (May 2007): *Star Cluster Holds Midweight Black Hole, VLA indicates* https://www.nrao.edu/pr/2007/globularbh/

[5] Toomre, A., Toomre, J., (1972): *Galactic Bridges and Tails* http://adsabs.harvard.edu/abs/1972ApJ...178..623T

[6] Moore, G., (April 1965): *Cramming more components onto integrated circuits*, Electronics, Volume 38, Number 8 https://drive.google.com/file/d/0By83v5TWkGjvQkpBcXJKT1I1TTA/view

[7] Plummer, H. C., (March 1911): *On the problem of distribution in globular star clusters* http://adsabs.harvard.edu/abs/1911MNRAS..71..460P

[8] M. Lauter, Toomre, A., Toomre, J., (remastered 2007) *Toomre & Toomre - Galactic Bridges and Tails (1971-1973) - film* https://www.youtube.com/watch?v=zvXLWEd2ZGU

[9] Jordan, M., (April 2010): *Berkeley "STAT260" Module: Monte Carlo Sampling lecture* https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture17.pdf

[10] George Djorgovski [**may be inaccurate**], (2004): *Relaxation Time verbal definition, caltech astrophysics* http://www.astro.caltech.edu/~george/ay20/Ay20-Lec15x.pdf

[11] Hut, P., Makino, J., (March 2002): *Readme for "nbody_sh1.C"* https://www.ids.ias.edu/~piet/act/comp/algorithms/starter/readme.html

[12] Hut, P., Makino, J., McMillan, S., (1995): *N-body Starter Code* https://www.ids.ias.edu/~piet/act/comp/algorithms/starter/nbody_sh1.html

[13] Hut, P., Makino, J., Teuben, P., (September 2007): *The Art of Computational Science, the Kali Code* The Plummer Model, pages 15-16, page 39 http://www.artcompsci.org/kali/vol/plummer/volume9.pdf

[14] F. Summers, NASA, ESA, (April 2008): *Hubble Studies Different Stages in the Collision Between Galaxies (Visualization)* http://hubblesite.org/video/578/news_release/2008-16

Appendix A Plummer Density derivation

Poisson's equation is used to convert between potential (ψ) and density (ρ):

$$\nabla^2\psi = -4\pi G\rho(r) \tag{5}$$

For equation (1), this is best done in spherical coordinates, as it is r dependent. So $\nabla^2\psi$ becomes:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\psi\right)$$

The working from here:

$$\frac{d}{dr}\left(r^2\frac{d}{dr}\psi\right) = 4\pi G\rho(r)$$

Rearranged to give:

$$\frac{d^2}{dr^2}\psi + \frac{2}{r}\frac{d}{dr}\psi = 4\pi G\rho(r)$$

where:

$$\frac{d}{dr}\psi = GM\frac{r}{(r^2+a^2)^{3/2}}$$

and:

$$\frac{d^2}{dr^2}\psi = GM\frac{-2r^2+a^2}{(r^2+a^2)^{5/2}}$$

Now, rearranging and combining with $\rho(r)$ on the L.H.S gives us:

$$\begin{aligned}
\rho(r) &= \frac{1}{4\pi G}\left[\frac{d^2}{dr^2}\psi + \frac{2}{r}\frac{d}{dr}\psi\right] \\
&= \frac{M}{4\pi}\left[\frac{-2r^2+a^2}{(r^2+a^2)^{5/2}} + \frac{2}{(r^2+a^2)^{3/2}}\right] \\
&= \frac{M}{4\pi}\left[\frac{-2r^2+a^2}{(r^2+a^2)^{5/2}} + \frac{2r^2+2a^2}{(r^2+a^2)^{5/2}}\right] \\
&= \frac{M}{4\pi}\left[\frac{3a^2}{(r^2+a^2)^{5/2}}\right]
\end{aligned}$$

Which can also be written in the form:

$$\rho(r) = \left(\frac{3M}{4\pi a^3}\right)\left(1+\frac{r^2}{a^2}\right)^{-5/2}$$

(Derivation written with help from [13])

Appendix B sph_gencluster.c

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include <string.h>

/*        This is the single cluster generation program for my N-body report.
        1) Asks user for:
                i)                  name of output file
                ii)         number of particles to generate,
                iii)         value of "a" for the plummer density
        2) Calculates mass of each particle, and after opening a file, puts no. of particles
                then mass at the start of the file.
        3) Generates the position of a particle in spherical coords.
        4) Accepts or rejects this point using the plummer density.
        5) If point is accepted, it's velocity is calculated using v_calc().
        6) Position and velocity are converted to cartesian, and put into file.

        The author understands that "LOOP: do..." is bad practice,
                but the program works anyway...

        Author: Matthew Rayner
*/

//-------------------------------------------------------------------------PROTOTYPES
double v_calc(double r);  //Calculates magnitude of velocity for point with length r
double rand_gen();                  //Generates random number between 0 and 1


int main()
{
//-------------------------------------------------------------------------DEFINITIONS
        int particles;
        double mass;
        double a;
        double x,y,z;
        double vx,vy,vz;
        double r,theta,phi;
        double v_mag, v_theta, v_phi;
        double plum;
        int i = 0;
        char file_name[50];
        FILE *sphere_cluster;
//-----------------------------------------------------------------------------RNG SEED
        srand(time(NULL));
//-----------------------------------------------------------USER INPUT/MASS CALCULATION
        printf("Please enter output filename (include .txt, less than 50 chars)\n");
        gets(file_name);
        printf("No. of Particles?: ");
        scanf("%d", &particles);
        printf("\nValue of a?: ");
```

```
        scanf("%lf", &a);


        sphere_cluster = fopen(file_name, "w"); //Opens file, checks if successful etc.
        if(sphere_cluster == NULL) {
                printf("Error creating output file, closing program\n");
                return 1;
        }

    mass = 1/((double) particles);              //Mass of each particle calculated

        fprintf(sphere_cluster, "%d\n", particles);         //Particles put in file
    fprintf(sphere_cluster, "0\n");
//-------------------------------------------------------------------POSITION CALCULATION
        LOOP:do{
                r = pow(rand_gen(), 1.0/3.0);

                plum = (3.0/(4.0*M_PI*pow(a,3.0)))* pow( 1 + ((r*r)/(a*a)) , -5.0/2.0);

                if( plum < pow(rand_gen(), 1.0/3.0) )
                                //restarts loop if plum < random no between 0 and 1
                        goto LOOP;          //i.e.  f(x) < Ag(x) ---> MONTE CARLO rejection
                else {
                        v_mag = v_calc(r);      //else computes points and velocities,
                                                //and prints into file
                        phi = 2 * M_PI * rand_gen();
                        theta = acos( (2 * rand_gen()) - 1 );
                        v_phi = 2 * M_PI * rand_gen();
                        v_theta = acos( (2 * rand_gen()) - 1 );

                        x = r * sin(theta) * cos(phi);
                        y = r * sin(theta) * sin(phi);
                        z = r * cos(theta);
                        vx = v_mag * sin(v_theta) * cos(v_phi);
                        vy = v_mag * sin(v_theta) * sin(v_phi);
                        vz = v_mag * cos(v_theta);

                        fprintf(sphere_cluster, "%0.9g %0.9g %0.9g %0.9g %0.9g %0.9g %0.9g\n",
                            mass, x, y, z, vx, vy, vz);
                        printf("Point %d accepted\n", i);
                        i++;
                }
        } while (i < particles);
//-------------------------------------------------------------------SUCCESSFUL PROGRAM END
        printf("\aGeneration \a complete, \aexiting program.");
        return 0;
}


//-------------------------------------------------------------------------RNG Function
// generates random point between 0 and 1, returns as a double
double rand_gen() {
        return (double)rand()/(double)RAND_MAX;
}
```

```
//-------------------------------------------------------------------------v_mag Function
//Uses a second Monte Carlo rejection technique to generate a velocity for each point
double v_calc(double r) {
        LOOP: do {
                double c, d;

                c = rand_gen();
                d = 0.1 * rand_gen();

                if( (c*c)*pow( (1 + (c*c)), 7.0/2.0) < d)
                        //Restarts loop if conditions for c are not met
                        goto LOOP;
                else
                        return (c * sqrt(2) * pow((1+ (r*r)), -1.0/4.0));
        } while ( 1==1 );
}
```

Appendix C splitter2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

/*      This program takes the data from the gravitylab output, and splits it into
        individual snapshots of the form "snap_001.txt" etc.
        NOTE: Please use a file with ONLY x,y,z coords (not the full simulation data)
                i.e. use awk to create a new file with the 2nd, 3rd and 4th columns only.
        1) Asks user for:
                i)          name of file to split
                ii)          number of particles in data set
                iii) output interval for data (no. of outputs per time unit)
                iv)          total simulation time
        2) Calculates:
                i)          size of each set ("set_size") (no. of elements in each set)
                ii)  number of "snapshots" ("total_snaps")
                iii) number of elements ("data_num")
        3) Allocates memory (dynamically-ish) to "mem_ptr" based on size of input file
        4) Copies data from input file to "mem_ptr" as a stream of elements
        5) (Hopefully) transfers correct elements to an individual "snap" for each time step

        WARNING: only use integer values when requested. Attempting otherwise will results
        in program failure. Especially for 1)iii), suggested values are 10, 20, 25, 50, or 100
        which correlate with an nbody_sh1.C simulation using -d 0.1, 0.05, 0.04, 0.02, or 0.01
        respectively

        Author: Matthew Rayner
*/

int main()
{
//-------------------------------------------------------------------------DEFINITIONS
```

```
        FILE *input_ptr;
        FILE *output_ptr;
        int i, j;
        int particles, sim_time, out_intr, data_num;
        int set_size;
        int total_snaps, snap_request, snap_start, snap_end;
        char filename[50];
//-------------------------------------------------------------------------USER INPUT
        printf("Enter filename to split up: ");
        gets(filename);
        printf("Enter number of particles: ");
        scanf("%d", &particles);
        printf("Enter output interval (outputs per time unit): ");
        scanf("%d", &out_intr);
        printf("Enter total sim time: ");
        scanf("%d", &sim_time);
        printf("\n");

//---------------------------------CALCULATE SET SIZE, TOTAL SNAPS, TOTAL ELEMENTS
        set_size = particles * 3;
        total_snaps = sim_time * out_intr;
        data_num = set_size * total_snaps;

//-------------------------------------------------ALLOCATE MEMORY FOR ELEMENT ARRAY
        double *mem_ptr;
        mem_ptr = malloc(data_num * sizeof(double));

        if(mem_ptr == NULL) {
                printf("ERROR! memory not allocated correctly.");
                return 1;
        }

//-------------------------------------------------OPEN INPUT FILE, COPY TO mem_ptr
        input_ptr = fopen(filename, "r");
        if( input_ptr != NULL ) {
                for(i = 0; i < data_num; i++)
                        fscanf(input_ptr, "%lg", &mem_ptr[i]);
                fclose(input_ptr);
        }
        else {
                printf("Error: Cannot find input file, ending program...\n");
                return 1;
        }

//-------------------------------------------------CREATE AND POPULATE "SNAP" FILES
        for(i = 0; i < total_snaps; i++) {
                char out_filename[sizeof "snap_1000.txt"];
                sprintf(out_filename, "snap_%d.txt", i);
                output_ptr = fopen(out_filename, "w");

                snap_request = i;
                snap_start = (snap_request * set_size);
                snap_end = snap_start + (set_size - 1);
```

```
                for(j = snap_start; j <= snap_end; j++) {
                        fprintf(output_ptr, "%0.9g ", mem_ptr[j]);
                        if(j % 3 == 2)
                                fprintf(output_ptr, "\n");
                }

                fclose(output_ptr);
        }

        free(mem_ptr);
        return 0;

}
```

Appendix D gif_setup.awk

```
set terminal gif animate delay 1
set output "animate_RENAME.gif"
set xrange [-3:3]
set yrange [-3:3]
set zrange [-3:3]
set view equal xyz
set ticslevel 0
```

Appendix E gif uploads

The author is aware that only the report itself can be marked, but would still like to present the animations he has made, in case they are of merit outside of a marking scheme: Please follow https://imgur.com/a/ayMCGTS for a full labelled list of simulations in real and velocity space (side by side).