Project 4: N-body simulation of an open galactic cluster

Joshua Bradt

(Dated: April 29, 2016)

A code was developed to model the motion of stars in an open galactic cluster using Newtonian gravity and the Runge-Kutta method of numerical integration. The calculation began with a uniform distribution of stars and found that the system quickly collapsed into a smaller region of space before expanding outward again. The system approached equilibrium toward the end of the calculation, but did not quite reach it. After discussing these results, some possible improvements to the calculation are proposed.

I. INTRODUCTION

An open galactic cluster is a group of similar-mass stars bound together by the gravitational force. These are interesting to astronomers since they provide a well-defined environment for studying the processes of star formation. [1] For the purposes of this paper, these clusters also serve as a simple yet physically relevant many-body system that can be easily simulated using basic Newtonian mechanics.

II. METHODS

As described in the report for Project 3, a system of N bodies interacting via the gravitational force can be described in Cartesian coordinates using the $3N$ coupled second-order ordinary differential equations [2]

$$\begin{aligned}
\frac{d^2x_k}{dt^2} &= \sum_{i\neq k} \frac{Gm_i}{r_{ik}^3}(x_i - x_k) \\
\frac{d^2y_k}{dt^2} &= \sum_{i\neq k} \frac{Gm_i}{r_{ik}^3}(y_i - y_k) \\
\frac{d^2z_k}{dt^2} &= \sum_{i\neq k} \frac{Gm_i}{r_{ik}^3}(z_i - z_k)
\end{aligned}$$

where

$$r_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (z_i - z_k)^2}.$$

Whereas in Project 3 each of these bodies represented a planet orbiting the Sun, in this case, each represents a single star or a small group of stars [1].

Once the equations above are established, the system can be solved using a variety of different methods. One of the most stable [2] and easily implemented [1] methods is the fourth-order Runge-Kutta method (RK4), which is described below: (reproduced from [2])

Algorithm 1 RK4 method for position and velocity

```
function RK4($\mathbf{r}$, $\mathbf{v}$, $h$)
    $\mathbf{k}_1^v \leftarrow$ findAcceleration($\mathbf{r}$)
    $\mathbf{k}_1^r \leftarrow \mathbf{v}$
    $\mathbf{k}_2^v \leftarrow$ findAcceleration($\mathbf{r} + \frac{h}{2}\mathbf{k}_1^r$)
    $\mathbf{k}_2^r \leftarrow \mathbf{v} + \frac{h}{2}\mathbf{k}_1^v$
    $\mathbf{k}_3^v \leftarrow$ findAcceleration($\mathbf{r} + \frac{h}{2}\mathbf{k}_2^r$)
    $\mathbf{k}_3^r \leftarrow \mathbf{v} + \frac{h}{2}\mathbf{k}_2^v$
    $\mathbf{k}_4^v \leftarrow$ findAcceleration($\mathbf{r} + h\mathbf{k}_3^r$)
    $\mathbf{k}_4^r \leftarrow \mathbf{v} + h\mathbf{k}_3^v$
    $\mathbf{v}' \leftarrow \mathbf{v} + \frac{h}{6}(\mathbf{k}_1^v + 2\mathbf{k}_2^v + 2\mathbf{k}_3^v + \mathbf{k}_4^v)$
    $\mathbf{r}' \leftarrow \mathbf{r} + \frac{h}{6}(\mathbf{k}_1^r + 2\mathbf{k}_2^r + 2\mathbf{k}_3^r + \mathbf{k}_4^r)$
    return ($\mathbf{r}'$, $\mathbf{v}'$)
end function
```

At each time step, this algorithm can be used to update the position and velocity of each particle.

One problem that can arise in this calculation is numerical instability when two particles move close together, causing $r_{ik} \to 0$. One way to prevent this is to add a small constant to the gravitational force to prevent the denominator from vanishing. This leads to an expression of the form

$$\mathbf{F}_{ik} = \frac{Gm_i m_k}{r_{ik}^3 + \epsilon^3}\mathbf{r}_{ik} \tag{1}$$

for the force vector on each particle.

III. RESULTS

I developed a program to simulate this system by repurposing the code from the previous project. [3] A system of 500 interacting particles was simulated with a normal initial mass distribution of mean mass $10m_\odot$ and standard deviation $1m_\odot$. The initial positions were uniformly distributed inside a sphere of radius 20 ly, and the particles started at rest. The simulation progressed with a time step of 0.1 Ma.

Before performing calculations, a value for the parameter ϵ in (1) needed to be found. The value $\epsilon = 0.1$ ly was found by trial and error to be the smallest value that prevented the output from containing discontinuities in particle kinetic energy. This value was used in the remaining calculations discussed in this section.

Three snapshots of the particle distribution are shown in Figure 1. The top image shows the initial conditions,

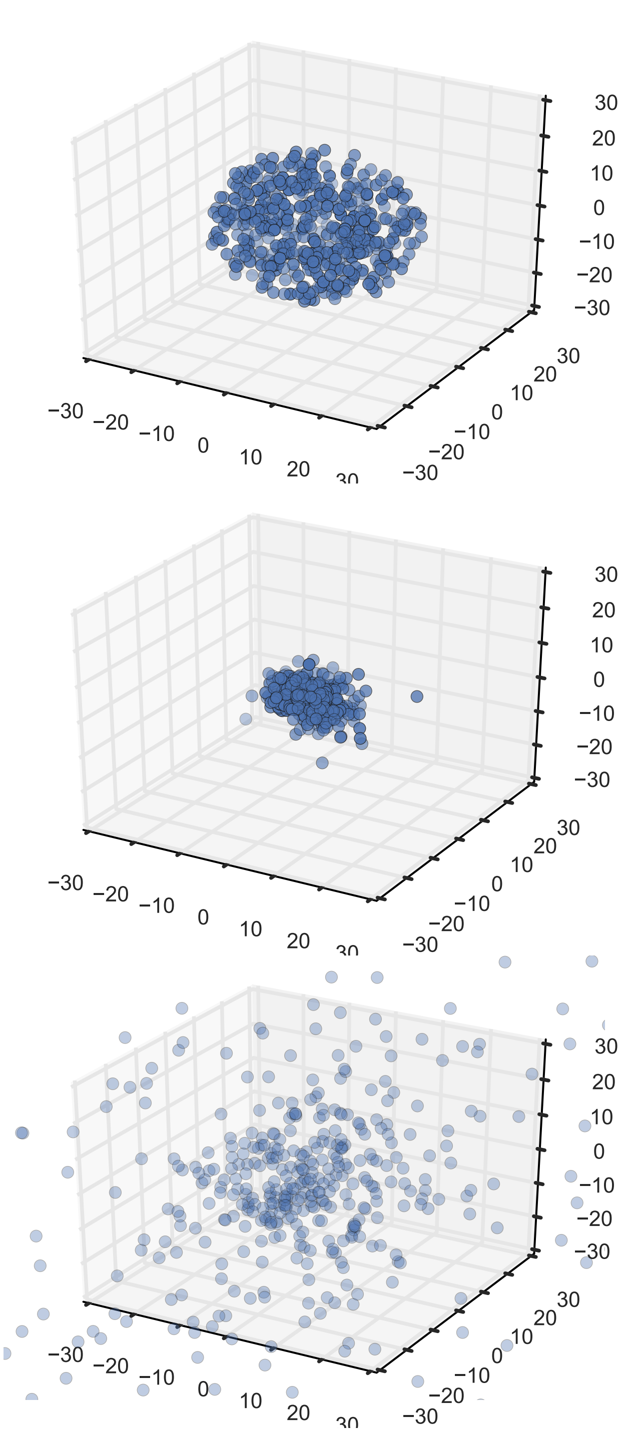

FIG. 1. Snapshots of the particles' positions at $t = 0\,\mathrm{Ma}$ (top), $t = 5\,\mathrm{Ma}$ (middle), and $t = 199.9\,\mathrm{Ma}$ (bottom). The particles begin uniformly distributed over a sphere, collapse into a smaller volume, and then scatter as some are ejected and others reach an equilibrium.

which were uniformly distributed over a sphere. The middle image shows that the particles initially move inward and collapse to a smaller volume. This collapse is also visible in Figure 2, which shows the radial positions of the particles as a function of time. Finally, the particles expand back outwards.

The energy of the system is shown in Figure 3 as a function of time. This plot only includes contributions from particles that remained gravitationally bound in the system for the duration of the simulation. Ejected particles identified by looking for particles whose total energy became positive at any point during the simulation. As shown in the figure, the system appears to approach equilibrium after around 50 Ma. However, the total energy does appear to be asymptotically approaching zero, which may indicate that particles continue to be ejected even at large times.

According to the virial theorem, if a system is in equilibrium, then

$$2\langle T\rangle + \langle U\rangle = 0, \tag{2}$$

where the angle brackets indicate a time average, or, by the ergodic hypothesis, an ensemble average. [1] Figure 4 shows a plot of this function over time for the bound particles. The curve approaches zero at large times, but never quite reaches it. This indicates that the system is approaching equilibrium, but has not quite reached it yet.

The number density of the particles is plotted in Figure 5 as a function of radial coordinate for the last simulated time step, time $t = 199.9\,\mathrm{Ma}$. The radial coordinate, in this case, is relative to the center of mass of the distribution of bound particles, rather than relative to the initial center of the distribution. This correction was necessary because the center of mass of the system drifted over time as particles were ejected. The distribution was fit using the function

$$n(r) = \frac{n_0}{1 + \left(\frac{r}{r_0}\right)^4}, \tag{3}$$

resulting in the parameters $r_0 = 15.09\,\mathrm{ly}$ and $n_0 = 8.623 \times 10^{-3} ly^{-3}$. The fit is quite good outside of approximately $r = 10\,\mathrm{ly}$, but the distribution does not follow the fit function closer to the center of mass.

IV. CONCLUSION

A code was written to simulate the motion of stars under the influence of gravity in an open galactic cluster. The forces on the stars were calculated using a simple Newtonian model of gravity. The results show the simulated particles initially collapsing into a smaller region of space before expanding outward again and achieving something close to equilibrium.

There are many clear improvements that could be made to this code. Most importantly, the model used

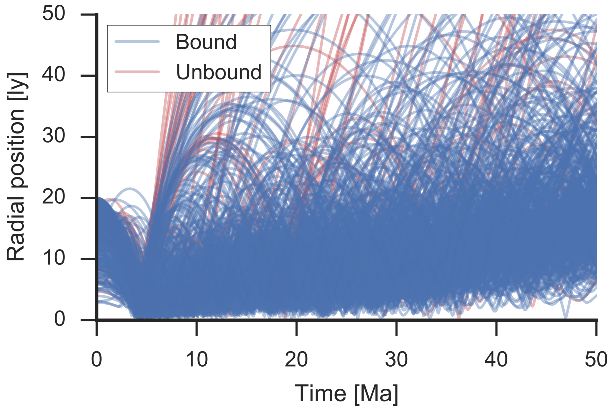

FIG. 2. Radial positions of the simulated particles.

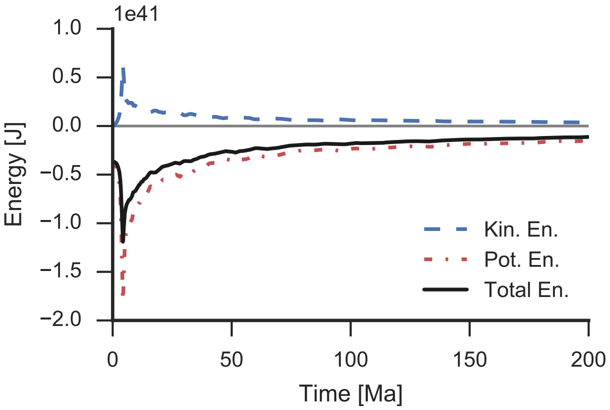

FIG. 3. Kinetic, potential, and total energy of the gravitationally bound particles as a function of time.

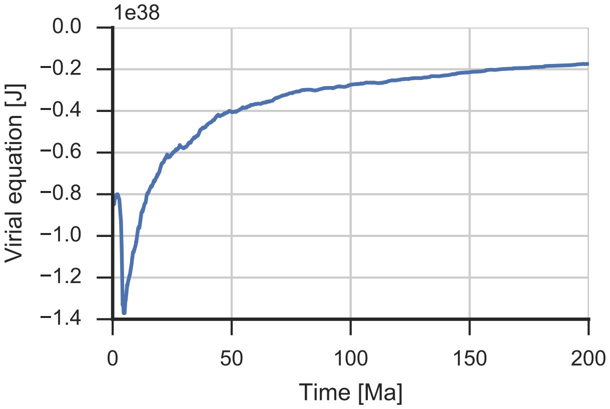

FIG. 4. Plot of the quantity $2\langle T\rangle + \langle U\rangle$ over time, which should equal zero when the system is in equilibrium if the virial theorem holds.

completely neglects relativistic effects, which is likely a poor approximation to reality when dealing with such massive objects. A more sophisticated model might also take the extent of the stars into account instead of treating them as point-like particles. Additionally, the algorithm could possibly be made more efficient, at least with respect to memory access, by representing the cluster of particles as a contiguous data structure like a matrix rather than a disparate set of objects scattered around in memory. The computations could also likely be parallelized without too much effort, thereby speeding up the code and increasing the number of particles that can be handled.

[1] M. Hjorth-Jensen, "Computational Physics MSU," (2016), https://github.com/CompPhysics/ComputationalPhysicsMSU.

[2] J. Bradt, "Project 3: A model of the solar system," (2016), https://github.com/jbradt/comp-phys/blob/master/project3/writeup/project3.pdf.

[3] J. Bradt, "Comp-phys repository," (2016), https://github.com/jbradt/comp-phys.

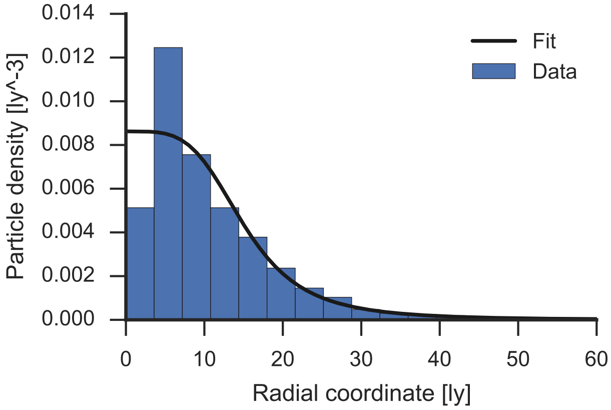

FIG. 5. Number density of particles as a function of the radial distance from the system's center of mass at time $t = 199.9\,\mathrm{Ma}$. The fit shown is the function $n(r) = \frac{n_0}{1+(r/r_0)^4}$ where $r_0 = 15.09\,\mathrm{ly}$ and $n_0 = 8.623 \times 10^{-3} ly^{-3}$.