Collisionless N-Body Simulation of System Formation

Computational Physics (PHY 391)

Tai Jespersen

Franklin & Marshall College

1.1 Abstract

I develop a collisionless N-Body simulation to analyze how evenly distributed massive systems evolve to stable systems and how the initial conditions affect the properties of those systems. In particular, I run a series of simulations to analyze how the initial angular velocity of a system affects the outcome. For each simulation, I measure each particle's approximate orbit and evaluate the average eccentricity over the entire simulation. I then observe and interpret the effects initial angular velocity has. My results indicate that the greater the initial angular velocity, the orbits become more circular, and greater in number. When the initial angular velocity passes a certain threshold, the particles get ejected from the system and the number of orbits decreases.

1.2 Introduction

During the beginning of the universe, matter was primarily Hydrogen and Helium gas distributed throughout space. Based on our current understanding, gravity was the primary factor in clumping matter together. Evidently, the gas did not collapse completely into a ball but instead formed stars, solar systems, and galaxies. The main reason why is that gas clouds have angular momentum, which causes the system to rotate. This rotation causes the gas to spin faster as it collapses closer to the center. In a 3D space, this means that mass within the axis of rotation forms a disk and mass that is not collapses. As the universe evolved, it eventually formed stars, solar systems, galaxies, and our planet. The result is a central body and a rotating disk of gas around it.[1] N-body simulations are commonly used in Astrophysics to simulate how bodies interact with each other, in particular due to the forces of gravity. Inspired by the question of how mass clumps together in a gas-like state such as that of the origin of the universe, I programmed a collisionless N-body simulation to visualize and quantify how different conditions affect the

outcome of this clumping of matter. After primary observation, the question became how these particles eventually form stable systems.

1.3 Theory

Newtonian Gravity is classically calculated using the formula:

$$F_i = \sum_{j=1}^{N} G\frac{m_i m_j}{r^2} \tag{1}$$

where r is the distance between masses m_i and m_j, and G is the gravitational constant. In N-body simulations, this is also referred to as the "Brute Force Method". The velocity can be calculated by the basic kinematic equation:

$$v_f = v_i + at \tag{2}$$

or

$$v_f = v_i + (\frac{F}{m})t$$

Where m is the mass and F is the gravitational force acting on the particle. The position can be calculated using the equation:

$$x_f = x_i + vt \tag{3}$$

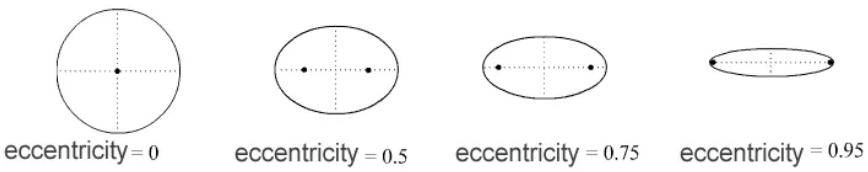

Figure 1: A visual representation of what ellipses with different eccentricities look like.

Ellipses are defined as a circular or oval shape with two focal points or foci where every point on the ellipse must have the same sum of the distance from both foci. The major axis is defined as the longest diameter or width of the ellipse and the semi-major axis is defined as half of that. The minor axis is defined as the shorted diameter, or width, of the ellipse and the semi-minor axis is defined as half of that. Ellipses also have a value called eccentricity as seen in Figure 1 calculated as:

$$e = \sqrt{1 - \frac{a^2}{b^2}} \tag{4}$$

Where a is the semi-major axis and b is the semi-minor axis.

An simple ellipse can be written using the equation:

$$Ax^2 + Bxy + Cy^2 = 1 \tag{5}$$

Or as a matrix of the quadratic form[3]:

$$A_{33} = \begin{bmatrix} A & \frac{B}{2} \\ \frac{B}{2} & C \end{bmatrix} \tag{6}$$

According to the principal axis theorem, we can solve for the semi-major and semi-minor axis of the ellipse by finding[3]:

$$a = \frac{1}{\sqrt{\lambda_1}} \text{ and } b = \frac{1}{\sqrt{\lambda_2}} \tag{5}$$

where a, the semi-major axis is related to the larger eigenvalue λ_1 of the matrix of the quadratic form of the ellipse. b, the semi-minor axis is related to the smaller eigenvalue λ_2 of the matrix of the quadratic form of the ellipse.

1.4 Computational Methods

Generation and Initial Conditions

The program works by generating a given N number of points on a grid in a circular pattern centered around the origin (0, 0). The initial x and y positions of each point are given by:

$$x = S \times \sqrt{R_V} \times cos(2\pi R_\theta) \tag{7}$$

$$y = S \times \sqrt{R_V} \times sin(2\pi R_\theta)$$

Where S is a given size, R_V and R_θ are random numbers generated from 0-1. The square root of R_V is taken to ensure the density of points is roughly equal regardless of how far from the origin the generated point is.

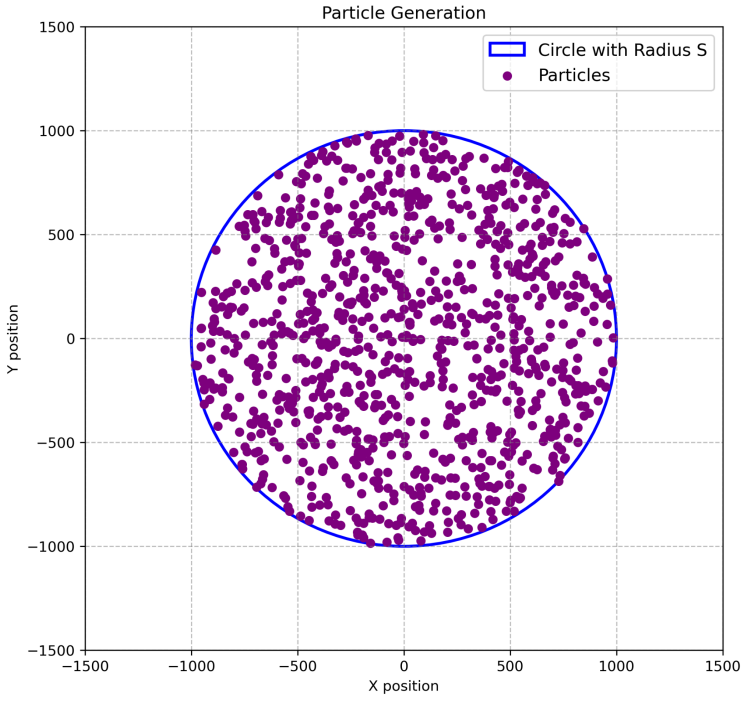

Figure 2: An example of generation where the Size S=1000 and N=1000 generated points. A circle with Radius S is also plotted to show the region the particle generation is bound to.

The points are also given a random mass M from 0 to an inputted value M_{max} and an initial angular velocity. The initial velocity for each point is given by:

$$v_x = -\alpha y \tag{8}$$

$$v_y = \alpha x$$

Where α is an inputted constant that scales the intensity of the initial angular velocity x and y and the initial positions in x and y respectively. In other words, the particle's initial velocity is tangent to its initial position vector.

Gravitational Forces and Velocity

The forces due to gravity are then calculated and the positions and velocities are updated accordingly. Because this simulation is collisionless, the Newtonian equation expressed in equation 1 breaks down for small values of r. Instead I use a softening equation given by:

$$F_i = \sum_{j=1}^{N} G\frac{m_i m_j}{r^2+\epsilon^2} \tag{9}$$

where ϵ is a given constant softening length or 100 in this simulation. The velocity and position of the particle is updated by using Equations 2 and 3 respectively. However, using this equation, the computing time becomes exponentially larger as the number of particles increases. This issue is tackled using the Barnes Hut Algorithm.

Barnes Hut Algorithm

I apply the Barnes Hut Algorithm to speed up the process of calculating gravitational forces for each point. Rather than calculate each force from each point individually, if a point is sufficiently far away from a group, we can use the center of mass of the group to calculate the force on the distant particle.

The Barnes Hut Algorithm divides points into groups and stores them in a quadtree. To start the space is divided into four equally sized quadrants called nodes. Particles are then inserted into the nodes and each node stores its center of mass. If a node meets certain conditions, in this simulation if there are more than two points and the node is larger than a set minimum size, the node subdivides into four child nodes and each point from that node is inserted into its corresponding child node. This process continues until no nodes can subdivide further.

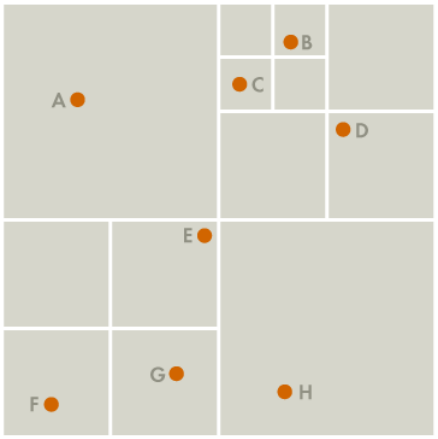

Figure 3: A visual representation of an example with 8 particles. Particles A through H are distributed randomly throughout the space, the grids represent different parts of the quadtree.[2]

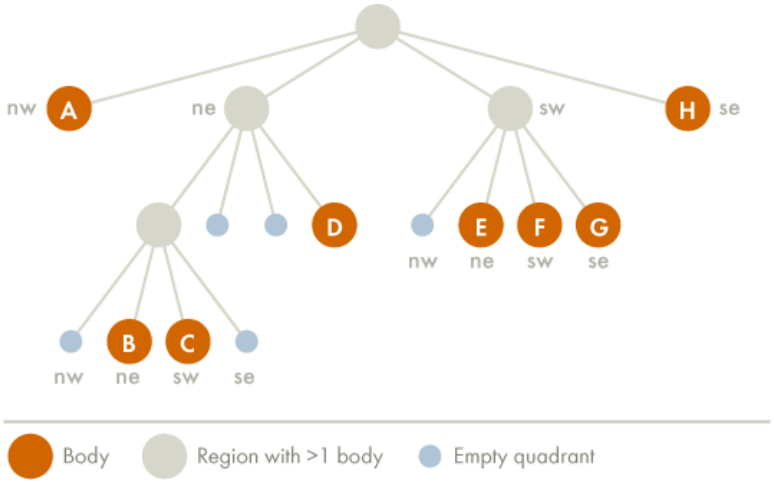

Figure 4: Another visual representation of the quadtree, where points A through H are stored in different nodes.[2]

To calculate forces, the quadtree is traversed, if at any point a node is sufficiently far away, meaning it meets the criterion that:

$$\frac{Node\ Width}{2 \times Distance\ from\ Particle} > \Theta$$ where Θ is set to 0.5 (The Barnes Hut Threshold)

The entire node and any child nodes underneath can be approximated using the center of mass for that node. This reduces the computing time for the gravitational forces. The "Brute Force Method" has a time complexity of $O(N^2)$, the Barnes Hut Algorithm reduces the time complexity to $O(N\ log(N)$.

Calculating the Eccentricity of Each Orbit

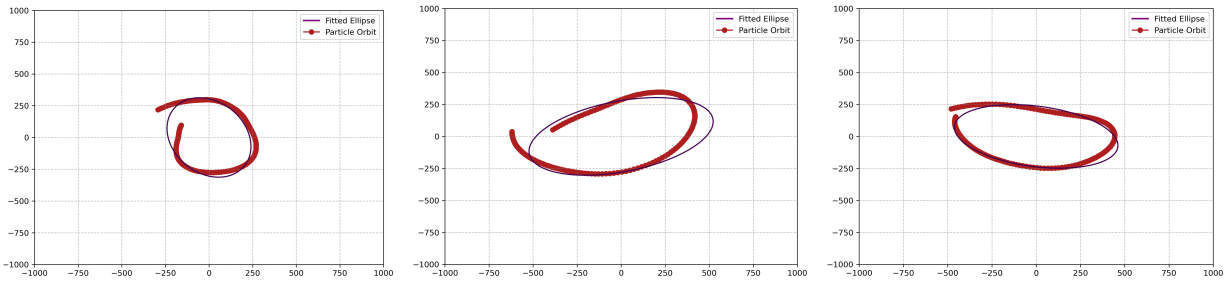

Figure 5: Plots of positional information from individual orbits in red, the fitted ellipse is plotted in purple.

In order to quantify how well the simulation is performing, I calculate an eccentricity for an ellipse fit to each particle's orbits as seen in Figure 5. For this simulation, a result that approximately resembles that of a solar system forming is an result where particles:

1) Orbit the center of Mass
2) Orbit with an elliptical orbit that does not approach an eccentricity of 1 (A line)

Each point starts off with an initial position (x,y) calculated in the point generation section. As its position is updated we can track it's angle θ using the formula:

$$\theta = tan^{-1}(\frac{y}{x})$$

For each orbit the initial angle θ_i is tracked, when θ is π radians away from θ_i it is marked as orbiting, when it returns to θ_i this set of points is counted as an orbit. If an orbit never leaves the central region of $-\frac{S}{10} < x < \frac{S}{10}$ and $-\frac{S}{10} < y < \frac{S}{10}$ the orbit is skipped. This is because the particle is considered as part of the center of mass and the orbit becomes too sporadic. This

can be seen in the videos referenced in the Appendix. If the orbit is counted, I begin by using the formula to solve for an ellipse of the form:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey = 1 \quad (10)$$

And use a fit function to solve for A, B, C, D, and E. Our first three terms of Equation 10, namely $Ax^2 + Bxy + Cy^2$ are important for calculating the semi-major and semi-minor axis of the ellipse, D and E affect the translation of the ellipse and can be neglected in our calculations. The matrix of the quadratic form can be represented as that of Equation 6. I can use the principal axis theorem to solve for the semi-major and semi-minor axis of the ellipse as seen in Equation 7, and then use those values to find the eccentricity calculated in Equation 4. These eccentricities are then plotted over time and the average eccentricity and total number of orbits are calculated.

1.5 Data and Analysis

By isolating one variable and keeping the rest constant, I am able to observe differences in the resulting outcome and numerical results. An important aspect of the end state of the system is the

initial angular velocity.

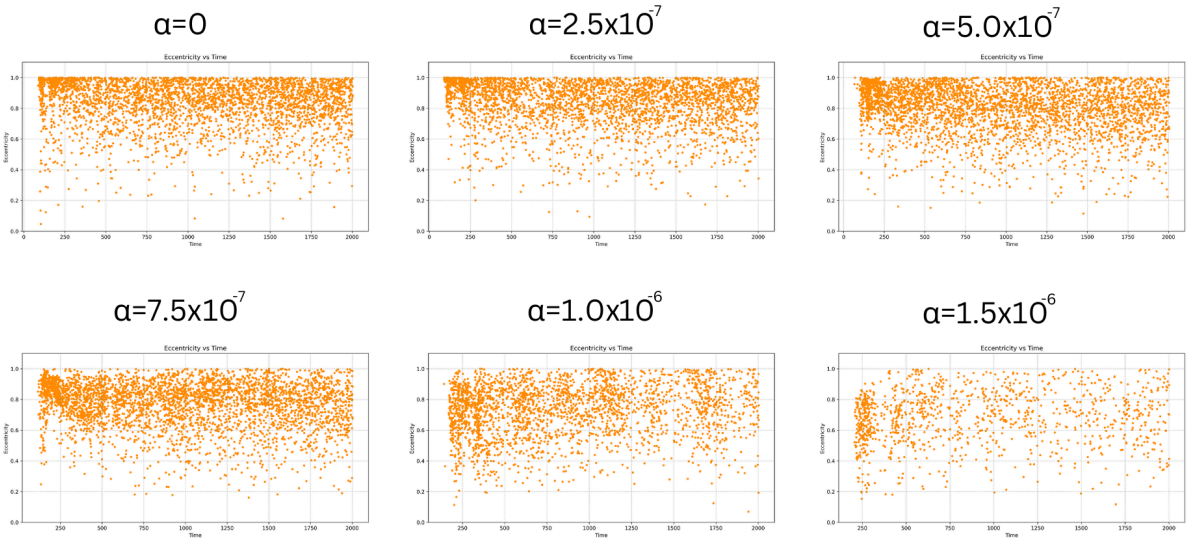

Figure 6: A plot graphing eccentricity over time for simulations with an initial constant of angular velocity of $\alpha = 0$, $\alpha = 2.5 \times 10^{-7}$, $\alpha = 5.0 \times 10^{-7}$, $\alpha = 7.5 \times 10^{-7}$, $\alpha = 1.0 \times 10^{-6}$, and $\alpha = 1.5 \times 10^{-6}$

I run simulations varying α and observe the results as seen in figure 6. For constants, The maximum mass (M_{max}) is set to 50000, the radius of generation (S) is set to 1000, the number of particles (N) is set to 1000, smoothing size (ε) is set to 100, the barnes hut threshold (Θ) is set to 0.5, and the simulations are run for approximately 33 seconds. The data trends toward lower eccentricities as α increases, and as α increases high enough the number of orbits drop off as seen in Figure 6. This is likely due to particles having an initial angular velocity that exceeds the gravitational force a particle has towards the center of mass and thus getting kicked out of the system entirely.

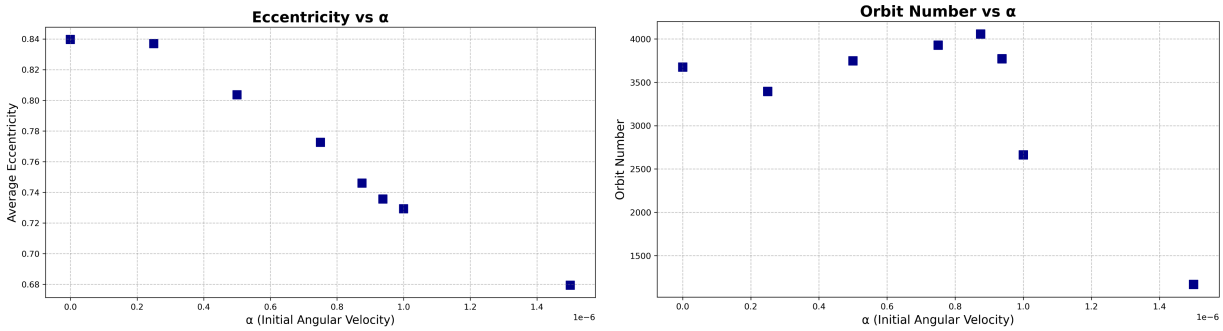

Figure 7: Plots graphing the average eccentricity and number of orbits vs α for 8 simulations with varying α values.

In another view of the results as seen in Figure 7, as α increases, the average eccentricity of the system decreases. This makes sense as a higher initial angular velocity will give the system a higher angular momentum and thus a more circular orbit. The orbit number trends upwards as α increases, until approximately $\alpha = 0.9$ where it then sharply decreases as seen in Figure 7. The initial increase is likely due to the angular velocity causing particles to fall towards the center less abruptly and subsequently getting ejected less. However, as the angular velocity approaches a value that is too high the particles are simply ejected from the system.

1.6 Conclusion

In conclusion, the results from my simulation indicate that the initial angular velocity does impact the dynamics of the system. In particular, as the initial angular velocity increases, the eccentricity decreases. The number of particles ejected from the system also decreases as angular velocity increases, until a certain threshold is reached where it abruptly increases. The simulation has a lot of potential to be expanded. Many other variables that were kept constant could be

modified and the results could be recorded. More data points could also be collected and a theoretical function could be developed and fit to the data.

1.7 References

[1]Star Formation. (n.d.). *Department of Astronomy, Case Western Reserve University.* Retrieved December 9, 2024, from http://burro.case.edu/Academics/Astr221/SolarSys/Formation/starform.html

[2]Ventimiglia, T., & Wayne, K. (n.d.). *The Barnes-Hut algorithm*. ArborJS. Retrieved December 9, 2024, from http://arborjs.org/docs/barnes-hut

[3]Evans, J. (2020, April 8). *Linear algebra 36: Eigenapplications, 2. Ellipses* [Video]. YouTube. https://www.youtube.com/watch?v=V5sxHXHZvjE

1.8 Appendix

Table of simulations used with videos linked for each test (A1)

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
Initial Angular Velo (α)	0	0.00000025	0.0000005	0.00000075	0.000000875	0.0000009375	0.000001	0.0000015
Max Mass	50000	50000	50000	50000	50000	50000	50000	50000
Generation Size (S)	1000	1000	1000	1000	1000	1000	1000	1000
Number of Points (N)	1000	1000	1000	1000	1000	1000	1000	1000
Total Time (s)	33	33	33	33	33	33	33	33
Smoothing Size (ε)	100	100	100	100	100	100	100	100
Barnes Hut Number	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Average Eccentricity	0.8396842878	0.8368947249	0.8035676713	0.7725503447	0.7460361803	0.735649507	0.7292035218	0.6793606297
Total Orbits	3674	3395	3748	3928	4056	3770	2661	1167

The simulation code is provided here:

Simulation Code (A2)