Simulating and Coliding Galaxies with Matlab

V. R. Whitney*
Department of Physics, Ithaca College, Ithaca, New York 14850

Using the program Matlab I generated a two-dimensional galaxy of particles interacting gravitationally. After generating and stabilizing the galaxy I placed it on a collision course with another galaxy of the same dimensions. This is a specific application of the well known and studied n-body problem, a mathematical model of n-number of bodies interacting gravitationally. The only way to solve this problem is analytically, which Matlab lends itself to well. The resulting simulations are the result of a series of programs I wrote to solve this model. The results of the simulation show a close correlation to systems observed in the Universe as well as other simulations made.

I. INTRODUCTION

The n-body problem is very well known within mathematics and physics. The problem consists of a number of bodies, n, interacting gravitationally. There is no simple solution to this problem, and can only really be solved analytically. There was a prize offered by King Oscar II in the late 1800s for the first person to find a solution for this problem. The prize ended up leading to the developement of chaos theory, based on the work of Poincare, but this solution did not actually answer the problem as stated. In order to actually answer the problem I will model the small motions of the bodies over increments of time by measuring the net acceleration on each object from the combined forces of every other object.

II. THEORY

Using the basic formula for force due to gravity

$$F_g = \frac{m_1 m_2 G}{r^2} \tag{1}$$

I calculate the force, or acceleration, that each particle will feel based on the presence of every other particle in the simulation for a given time step size, t. Then, in the t for-loop, I simply add up the accelerations applied by gravity and apply them to the velocity of each particle, then move them by one step, then repeat. These calculations are broken down into the following.

$$A_x = \frac{k\delta x}{r^3},\ A_y = \frac{k\delta y}{r^3}, because\ cos(\theta) = \frac{\delta x}{r} \tag{2}$$

Then I simply sum all of the accelerations on a single particle to get what the motion will be, save that in a vector, then calculate the motion of the next particle. Only once all of the particles' motions in a given dt are calculated will the program then change the location of the particles and then begin calculating new accelerations.

III. SIMULATION

Simulating a stable galaxy using the programs is very difficult, even for the simple case of equal masses. Getting the right constants for acceleration and initial velocity were difficult as they change in a non-linear way with the number of particles. At modeling the interactions the program is very effective, it is simply getting the right combination of constants within the program.

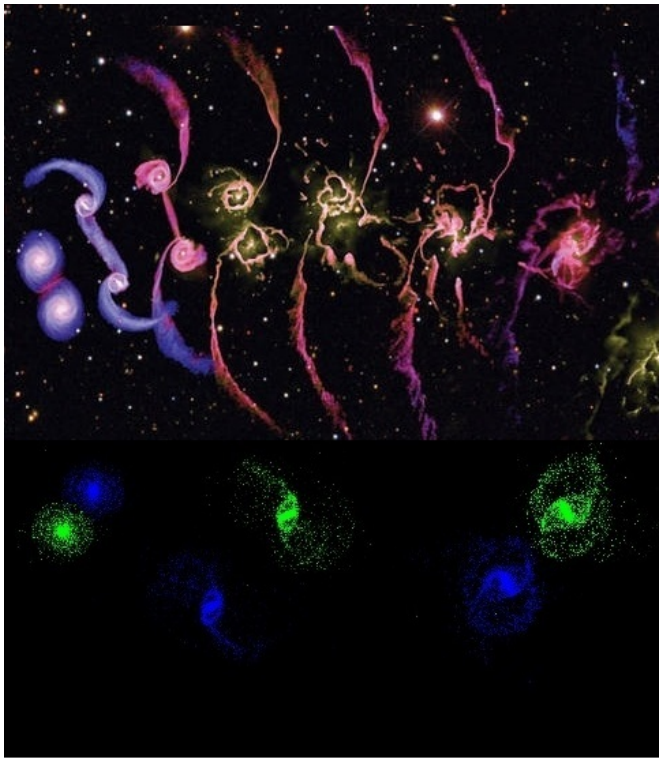

FIG. 1: Comparing the results of my simulation (bottom) to those of Tiziana Di Matteo (MPE/CMU), Volker Springel (MPE) and Lars Hernquist (Harvard).[2]

After generating a stable galaxy I then export a matrix containing all of the final coordinates of the galaxy, then place them as the initial conditions for the next simulation, which is the collision of two galaxies. I placed them on a slightly off-center collision course to show the particles thrown off associated with these types of galactic collisions.

The time required for the simulations increases with the number of particles squared. This is due to the fact

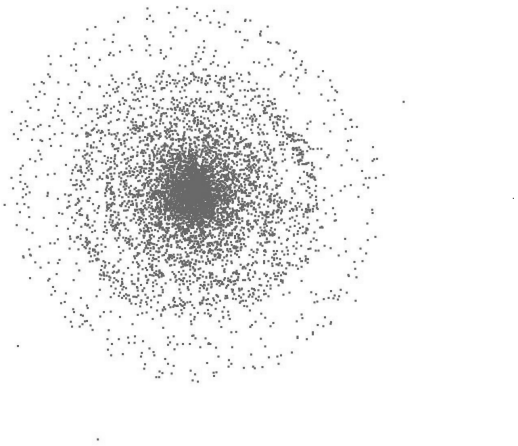

FIG. 2: A galaxy simulated with the program. After giving the particles initial conditions I allowed for a certain amount of settling time for the galaxy to stabilize.

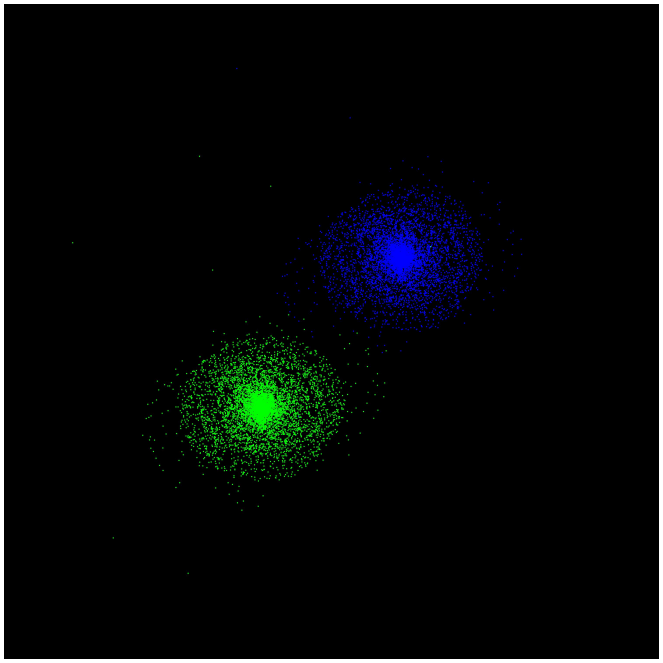

FIG. 3: Two galaxies of 7500 particles each, set on a collision course. The entire simulation took approximately 6 hours for 15000 particles, averaging 1 frame every minute.

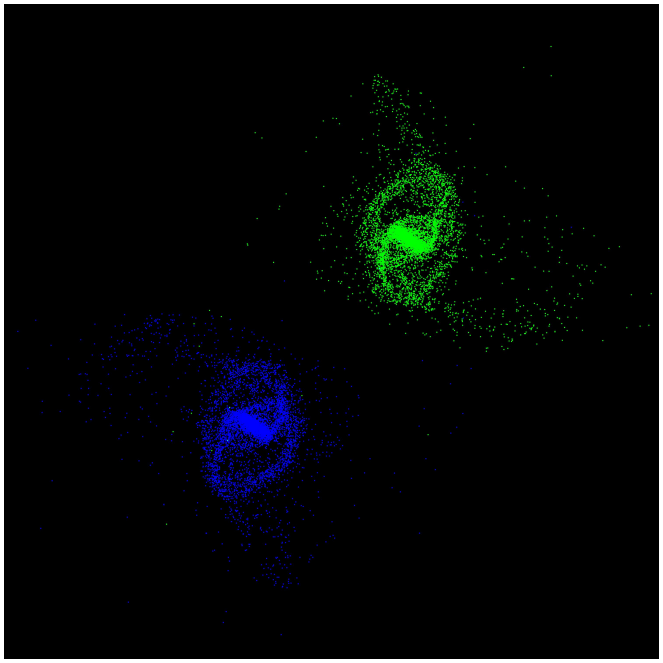

FIG. 4: A short time after the initial collision of the two galaxies. As you can see they are settling into bar galaxies before they collide again.

that the number of calculations also increases by the same factor. In order to generate a simulation I must save this individual frames as jpegs and then use Matlab to compile them into a movie after the simulation is done. Due to the amount of time that the calculations take the time step of my simulation is on the order of thousands of years for a galactic system. This proves problematic because as two particles pass each other they will essentially "jump" past one another, effectively gaining energy from ignoring a large portion of the negative work done on them from gravity. In order to try and reduce the effects of the large step size problem I truncate the accelerations for two particles if they would pass by each other in one step. I also set a minumum radius of $< \frac{1}{2}$ of the size of a pixel. What this effectively does is simulate the same motion for the majority of the simulation an overwhelming majority of the time. The only time that these conditions are tried is when interactions that would normally result in unrealistic motion are the alternative. These conditions also cause the energy in the system to be more concerved than they would be without them.

IV. ERROR

One problem with my simulation is that the time required to generate a single frame is proportional to the number of particles, N, squared. If I were to use the Barnes-Hut approximation[1], which calculates the center of mass for high density areas further away, then my simulation would scale as N*logN as opposed to . My simulation for 15,000 particles, which took 6 hours, could be reduced down to 6 seconds. Or, given the same amount

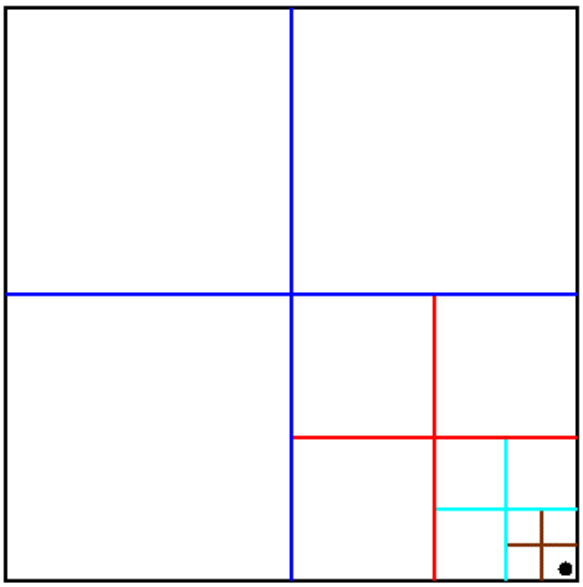

FIG. 5: Diagram depicting the Barnes-Hut approximation. This method uses approximates the force for particles farther away as proportional to the center of mass of those particles.

of rendering time, I could have calculated a simulation of 30 million particles, given equal time steps. Given the reduction of time of rendering per frame I would simply calculate 100 or more time steps in between frames to reduce the errors found. As the step size decreases the amount of times my program's unrealistic exceptions have to be made are decreased substantially, allowing for a realistic simulation with less exceptions.

V. CONCLUSIONS

My resulting simulations show a remarkable similarity to other accepted gravitational simulations of two galaxies interacting. Given the size of the time step I was still able to get fairly accurate simulations and for a sizable number of particles. Given more time I would definitely apply the Barnes-Hut approximation and reduce the step size to limit the chance of Matlab breaking conservation laws.

---

* Electronic address: vincewhitney@gmail.com

[1] *Barnes-Hut Approximation* JOSH BARNES and PIET HUT The Institute for Advanced Study, School of Natural Sciences, Princeton, New Jersey 08540, USA

[2] Tiziana Di Matteo (MPE/CMU), Volker Springel (MPE) and Lars Hernquist (Harvard).

[3] MATLAB numerical analysis software, <www.mathworks.com/>.