Final Report

PHYS349 - Advanced Computational Physics

David René 20806163

Department of Physics and Astronomy
Faculty of Science
Waterloo, Canada

1 Statement

As of the present time, I declare all work to be fully my own. The project will have a single independent model that will be developed by myself. In addition, I will work to develop robust and various tests, which I will use to evaluate both models. Finally, as a fun challenge, I will attempt to create a user interface, which will allow users to use pre-made initial configurations (such as the one presented as special cases) and see how these evolve through time.

2 Introduction

The gravitational n-body problem is applicable to a large set of systems, which is what makes models and simulation of this problem so valuable. While analytical solutions of the n-body gravitational problem don't necessarily exist, numerical solutions do exist but are tough to compute, which is why we use the computer. In such a system, we imagine n different bodies, all having a gravitational force onto the other. We assume no other forces are affecting these bodies and that they are point-like particles. The force applied on each individual body is described by the sum of the gravitational forces caused by all other bodies. Given a 0th body, with n other bodies, we can find the acceleration on this body:

$$\sum F = m_0 a_0$$
$$a_0 = \frac{1}{m_0} \sum_{i=1}^{n} F_{G,i}$$

Where $F_{G,i}$ is the gravitational force caused by the ith other body. We recall our equation for the gravitational force:

$$F_{G,i} = \frac{G m_0 m_i}{r_{0i}^2}$$

Where m_i is the mass of the ith particle, G is the gravitational constant and r_{0i} is the distance between the 0th body and the ith body. Substituting this into our original equation gives:

$$a_0 = \sum_{i=1}^{n} \frac{G m_i}{r_{0i}^2}$$

This is not exactly quite what we want: we have that this scalar is simply the norm of the acceleration, the vector of which is:

$$\boldsymbol{a_0} = \sum_{i=1}^{n} \frac{G m_i}{r_{0i}^2} \hat{\boldsymbol{r}}_{\boldsymbol{0i}}$$

Where $\hat{\boldsymbol{r}}_{\boldsymbol{0i}}$ is the unit vector in the direction from the position of the 0th particle to the position of the ith particle. This means each vector in this sum will have a different direction, making it hard to sum. In order to solve this problem, we can project each of these vectors onto the x-y-z axes by finding the Euler angles. This is in fact twice as useful as we expect because it is very helpful to do so when working with numerical integration and phase space derivative, which must be described in components. This is where our theory stands for now.

3 Algorithms

In terms of algorithms, I implemented a phase space derivative function which will allow us to find the instantaneous change in phase space of any particle given the whole system and given a specific time. With this in hand, we can then proceed with numerical integration. For integration, I implemented a single method of integration: Leapfrog. I noticed mistakes with how I previously implemented this algorithm in my assignment and I will spend a lot of time to ensure that it is implemented correctly and is efficient.

A major issue with these models is how they can become computationally expensive very fast. I did my best to implement code that is efficient such that my model runs in a timely manner.

In general, I used Object Oriented Programming to keep track of each property of our system as well as to easily connect to the user interface (if there is one, eventually)

4 Validation

I wrote a test suite, which tests both our acceleration and integration. These tests can be run by running the "./src/tests/test_v1.py" file, or by executing "./src/main.py" and clicking "Run the tests". All tests are logged within the "./logs" folder. Here is a standard output of those tests:

```
test_magnitude (__main__.TestAcceleration)
Tests the magnitude of the acceleration of the me on the Earth ... ok
test_orientation (__main__.TestAcceleration)
Tests the orientation of the acceleration of 2 particles aligned in the x axis .
test_3body (__main__.TestIntegration)
Tests the integration of 3 particles ... ok
test_acc (__main__.TestIntegration)
Tests the integration of a particle with acceleration ... ok
test_conservation_energy (__main__.TestIntegration)
Tests the conservation of energy in the system ... ok
test_no_acc (__main__.TestIntegration)
```

```
Tests the integration of a particle with no acceleration ... ok

----------------------------------------------------------------------
Ran 6 tests in 1.359s

OK
```

Which shows all test ran perfectly.

5 Results

6 Discussion

7 Conclusions

I apologize for the lack of content and the unfinished results. As mentioned in private messages to Prof. Hudson, my partner gave up on the course and "ghosted" me at the very last minute. I had been waiting for him to put in some work and I waited way too long, at which point I learned from someone else that he was abandoning the course. At that stage there was too much work to do in too few time for me to complete this project, and that is my fault.

While this report is not complete and the challenge I gave myself to create a GUI is also not complete, please have a look at the code I wrote: there are unit tests, which can be run from the GUI (only working functionality in the GUI). I also integrated Earth's orbit around the sun for a quarter of the year and made a movie of that ("./assets/quarter-year-orbit.mp4"). I wrote extensive documentation for the entirety of my code and the 2 study cases can be generated from the model constructor. I did not have time to animate these study cases properly or have a look at them.

Have a great summer and thank you for your time.