Interim Report

PHYS349 - Advanced Computational Physics

David René 20806163
Luong Dang 20794324

Department of Physics and Astronomy
Faculty of Science
Waterloo, Canada

1 Statement

As of the present time, we declare all work to be fully our own. The project will have two independent models that will be developed by both co-authors individually. In addition, we will work together to develop robust and various tests, which we will use to evaluate both models. Finally, as a fun challenge, David will attempt to create a user interface, which will allow users to visualize and edit the initial configuration, 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, we 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, we will implement 3 methods of integration: Leapfrog, RK4 as well as ODEInt by SciPy. The first will be great to highlight the problems with the second in physical scenarios, and the third will allow us to test the first two. I noticed mistakes with how I previously implemented these algorithms in my assignments and I will spend a lot of time to ensure that these are implemented correctly and are efficient.

A major issue with these models is how they can become computationally expensive very fast. We will implement code that is efficient such that our model runs in a timely manner.

In general, we will use 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

We have started writing some simple assertion based tests as we build our codebase in order to test out the most basic components of our code. These tests consists of the model being used and checking that what is done through our model is what should be expected, including all edge cases, such as two particles being on top of each other or a particle having non-real coordinates. In terms, we want to implement these tests for every feature and in addition, we want to compare our results with results produced by already existing models as well as results from real life.