PHY480 Project Title

Your registration number

December 2, 2018

Abstract

This LaTeX document serves as a template for the PHY480 Autumn Semester report, which is due for submission by the Friday of week 12. The report should have a short abstract, up to about 200 words long, which summarises the contents of the report.

1 Introduction

2 Literature Review

In order to undertake the project it was necessary to first inform myself of the N-Body problem, and its pertinence in Astrophysical problems. To do this, a literature review was undertaken.

2.1 The N-Body Problem

Consider a system of particles, or Bodies, in a dynamic system. In order to understand how the system will behave, the forces acting on each particle need to be ascertained, as well as how this force varies with time. In very simple systems, namely those involving only two particles, there is often an analytical solution to understand how it will behave. Such as the Bohr model of an atom. However, when multiple particles are introduced, more than three to be specific, the problem is impossible to solve analytically, without making simplifying assumptions[1]. Hence, the N-Body Problem.

To overcome this, the equations of motion governing the system are instead solved numerically. When the problem was first identified In the early 17th century, solving these equations to resolve long term dynamical behaviour was completely unfeasible. Consequently, it wasn't until the advent of computers that N-Body simulations began to proliferate in the Astrophysics community.

2.2 The N-Body Problem in Astrophysics

In Astrophysics the distance scales being considered are often much greater than those at which the electrostatic forces act, let alone the nuclear. As a result, the gravitational force is often the one that needs to be considered. Hence, astrophysical problems, that analyse the dynamical effects of gravity dominated systems, are the ideal candidate for N-Body simulations.

The latter stages of star formation is perhaps the smallest scale at which N-Body simulations become useful. In the former stages, particle sizes are very small, leaving millions of particles to account for, hence, fluid mechanics are a more computationally feasible way of modelling them. However, this changes once in the runaway and oligarchic growth stages: The onset of more significant gravitational forces, and decrease in number of bodies sees N-Body simulations become a much more effective method.

E. Kokubo and S. Ida were among the first to utilise showcase this, in their paper *On Runaway Growth of Planetesimals*[1]. They performed 3-D N-body simulations of 3000 equally massive planetesimals. The simulations provided conclusive evidence of naturally occurring runaway growth, something only previously inferred by statistical analysis of the coagulation condition[2, 3]. The results confirmed the validity of a statistical approach to demonstrate runaway growth. It also highlighted the disadvantages of the statistical approach in later stages of runaway growth, where the planetesimals are grouped spatially, as a statistical approach relies on a homogenous distribution of planetesimals.

[1] Such a Rydberg atom in atomic Physics that makes the assumption of a 'Hydrogenic' atom, simplifying the problem to the Bohr model.

3 Progress on Project

There are a many ways to approach the N-Body problem to the 4th order. More accurate methods are predictor-correctors, in particular the Hermite Polynomial, and the Adams-Bashforth-Moulton model. The former tries to predict the higher order kinematic components, the aptly named Jerk, Snap, Crack, and Pop. This is computationally expensive, as crossproducts between velocity and acceleration vectors are needed, something relatively inefficient for computer processors to calculate. The Adams-Bashforth-Moulton method predicts future values by fitting a polynomial to previous values, this requires storing previous values of the quantity that needs predicting.

At low values of N, storing these values is more computationally feasible hence, the method is more efficient than the Hermite Method, hence, this project is going to implement the Adams-Bashforth-Moulton model. This method is not self starting, meaning in order to function it requires time-history of the velocity and acceleration vectors. In order to do this, a second order method is necessary to initialise the values used for a time history, in a method known as 'Bootstrapping'. Hence, the first part of the project required implementing an accurate 2nd order N-Body.

3.1 Second order Kinematics

In order to calculate a body's changes in velocity and position over a given time-step, the acceleration vector acting on it must first be calculated. In a system dominated by gravitational forces, this requires knowledge of every body's position in a consistent coordinate system. For a given body, n, its interaction with another body m, can be calculated using Newton's Law of gravitation, provided the distance between them, $\vec{r_{nm}}$, is known. By summing over all interactions the body will have, its total acceleration vector can be calculated as shown in eq. (1).

$$\vec{a_n} = \sum_{m \neq n}^{N} \frac{Gm_m}{|\vec{r_{nm}}|^3} \vec{r_{nm}} \tag{1}$$

Given a discrete time-step over which to apply this acceleration, the changes in distance and velocity can be determined using the kinematic equations. These are shown below, where the subscripts i and f denote the initial and final positions respectively, δt represents the time-step, and $\vec{r}$,$\vec{v}$ and $\vec{a}$, the position, velocity and acceleration.

$$\vec{r}_{n,f} = \vec{r}_{n,i} + \vec{v}_n \delta t \qquad\qquad \vec{v}_{n,f} = \vec{v}_{n,i} + \vec{a}_n \delta t$$

For the position, this can easily be extended to the second order, by including the current acceleration. This is shown in eq. (2).

$$\vec{r}_{n,f} = \vec{r}_{n,i} + \vec{v}_n \delta t + \frac{1}{2} \vec{a}_n \delta t^2 \tag{2}$$

To calculate the velocity to the second order, the rate of change of the acceleration is required, the jerk. This is not a simple calculation, and is computationally expensive, hence, an approximation should be made: $\frac{da}{dt} \approx \frac{a_i - a_f}{\delta t}$. Given a small change in acceleration this in a valid assumption. Programatically, this requires storing the previous value of acceleration, a_p. Then by using it in conjunction with the current acceleration, a_c, the second order equation for the velocity can now be taken as eq. (3).

$$\vec{v}_{n,f} \;=\; \vec{v}_{n,i} + \vec{a}_n \delta t + \frac{1}{2} \frac{d\vec{a_n}}{dt} \delta t^2 \;=\; \vec{v}_{n,i} + \vec{a}_{n,c} \delta t + \frac{1}{2} (\vec{a}_{n,p} - \vec{a}_{n,c}) \delta t \tag{3}$$

3.2 Recreating the Solar System accurately

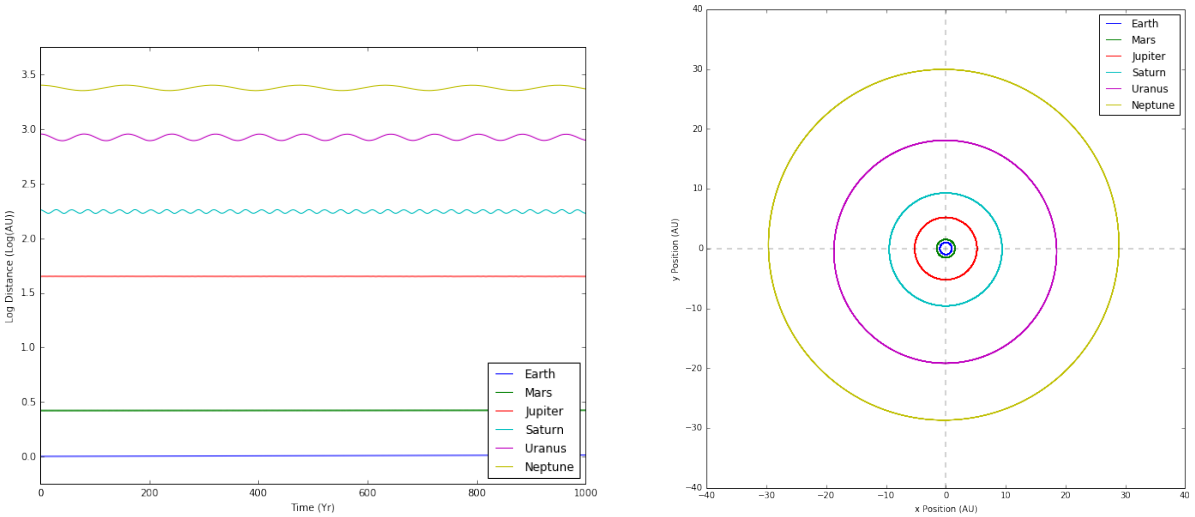
To undertake N-Body simulations, these are the only equations necessary. However, to start the process, initial conditions should be given to each body, namely a mass, position, and velocity. These were taken from a NASA database[4], see appendix A for the values pertinent in this project.

To initialise the simulation, the planets were positioned on the x axis at their orbital radii, and at $y = 0$. Their respective orbital velocities were then applied along the y axis, with $v_x = 0$. Hence, when the simulation was run they would orbit on the x–y plane.

At the beginning of each time-step, the acceleration acting on each body was calculated using eq. (1), this value was then applied to the kinematic equations, eq. (2), and eq. (3). The planets were moved to these positions, and given the corresponding velocities. New accelerations were calculated, and the

process was repeated. Since no time history was given, initially the change in acceleration couldn't be calculated in eq. (3). Hence, only the first derivative of velocity was used in the calculation of $\vec{v}_{n,2}$, this was done by manually setting the change in acceleration to zero.

To demonstrate the accuracy of this second order code, a 1000 year simulation of the solar system was carried out with 10 second time-steps. The resulting simulation produced a system with stable orbits, this is reflected in fig. 1. Figure 1 (a) shows the log of orbital distance against time, it is clear the orbits remain at a consistent distance with only Earth drifting very slightly. The oscillations in the distances of Saturn, Uranus, and Neptune, are due to the elliptical nature of their orbits, the eccentricities were reasonable: ∼0.02, ∼0.04, and ∼0.03 respectively. Figure 1 (b) shows the orbits in the x–y plane, demonstrating their circular nature.



(a) Orbital distances against time

(b) x-y position of orbits

Figure 1: Plots of orbital position throughout the simulation.

To further ensure the model's accuracy, energy conservation was considered. The total energy was calculated when the system was initialised to obtain E_i, this was simply the sum of the kinetic and potential energies, as shown in eq. (4). A factor of a half was applied to the potential energy, as in this summation it would be calculated twice for a given interaction $n \leftrightarrow m$.

$$E_{tot} = \sum_{n=1}^{N} \left[\frac{1}{2} m_n |v_n|^2 - \sum_{m \neq n}^{N} \frac{G m_m m_n}{2 |r_{nm}|} \right] = \frac{1}{2} \sum_{n=1}^{N} m_n \left[|v_n|^2 - \sum_{m \neq n}^{N} \frac{G m_m}{|r_{nm}|} \right] \tag{4}$$

This value was recalculated throughout the simulation, and compared against the initial value, using eq. (5). Where ΔE, represents the fractional energy loss, and the subscript t denoting the time-step at which the value is calculated.

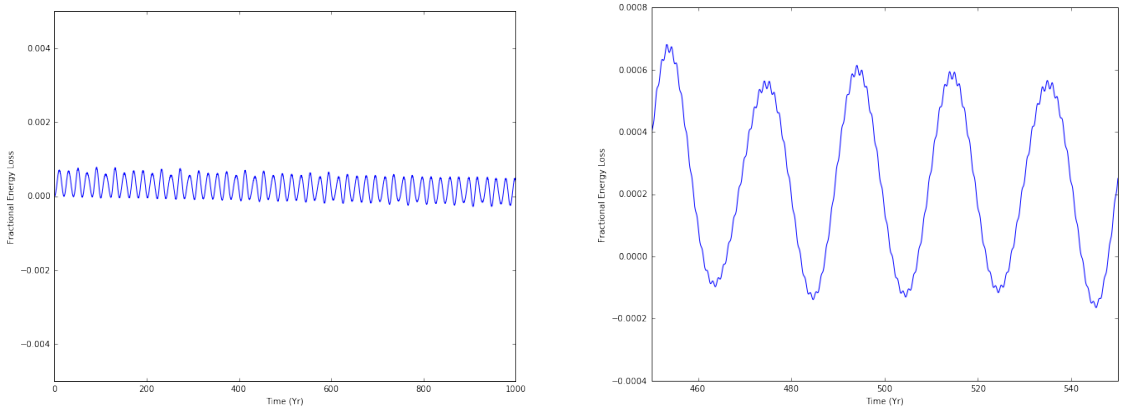
$$\Delta E_t = \frac{E_i - E_t}{E_i} \tag{5}$$

This fractional energy loss was plotted against time, to see how it varied throughout the simulation. This is shown in fig. 2. Figure 2 (a) shows the variation across the entire simulation, whereas fig. 2 (b) shows the value over a 100 year time period.

Over the whole simulation, the average energy loss remains close to zero, only lowering slightly, mainly due to the drift of Earth. However, there are clearly oscillatory fluctuations in the value, these oscillations are due to interactions of Jupiter with Saturn. Upon increasing the time-resolution of this graph, fig. 2 (b), smaller scale oscillations can be seen. These are presumably a result of interactions between the other planets in the system. Nonetheless, the energy losses (and its fluctuation) proved to be minor in this simulation; the large scale oscillation having an amplitude of ~ 0.0003, and only lowering by ~ 0.0002.

3.3 Suitability for Bootstrapping

The previous section demonstrates the code's ability to accurately recreate the orbits of the Solar System, to a reasonable degree of accuracy. However, if a simulation is to run for millions of years, the discrepancies



(a) Energy loss over simulation

(b) Energy loss over 100 year period

Figure 2: Plots showing the Energy loss throughout the simulation.

seen in the 2nd order system will manifest as large inaccuracies. Hence, the 4th order Adams-Bashforth-Moulton will be preferred.

As a method of producing the initial time history, the 2nd order code is more than sufficient; over a small timescale, < 80 seconds, the errors seen at long timescales will be infinitesimal. Accordingly, this code will be used initially in the 'bootstrapping' phase.

3.4 Randomising Initial Positions Along Orbit

If the simulation is to be run more than once, it will be beneficial to randomise the initial positions of the planets along their orbit. Should the code be applied to astrophysical problems, this would prevent any deterministic results from arising, thereby producing more natural behaviour.

There are many ways to approach this, some requiring more knowledge of the orbital paths than others. However, with knowledge of the mean orbital radius and velocity, the planet can be placed at any position at a radius equal to the orbital radius. Provided the velocity vector is both perpendicular to the radial vector, and equal to the mean orbital velocity.

Consider the positive quadrant of a planet's orbit (see fig. 1 (b)). On a given axis, the position will take a value between 0 and R, and the velocity will take a value between 0 and v_T, where R and v_T denote the orbital radius and velocity respectively. The component on the other axis can then be calculated using trigonometry. This is represented mathematically as shown in eq. (6), where X_n represents the random number between 0 and 1, assigned to the body, n.

$$r_{x,n} = X_n R_n \qquad \text{and} \qquad v_{y,n} = X_n v_{T,n} \tag{6}$$

Then by using Trigonometry, the components along the other axes can be calculated, as shown in eq. (7). Note the opposite axis on r_n, and v_n respectively in eq. (6), this is due to the velocity vector being perpendicular to the radius vector.

$$r_{y,n} = R_n^2 - r_{x,n}^2 \qquad \text{and} \qquad v_{x,n} = v_{T,n}^2 - v_{y,n}^2 \tag{7}$$

To allow for the other three quadrants, a random factor of ± 1 should be applied to each component of the position. The value applied to the velocity will depend on the value applied to the radius vector. That is to say, the quadrant allocated by the random factors of ± 1 will determine the value of ± 1 applied to the velocity components.

This was achieved programatically by first randomly assigning a value of ± 1 to each position vector component, a parity. Then, using a simple series of *IF* statements, the velocity components were allocated parities, thus creating two 1×2 arrays for each quantity. The conditions for the algorithm are illustrated in table 1.

Regardless of which quadrant they're in, the same trigonometric relations in eq. (7) exist between the quantities, and their components. Hence, with a random number allocated to each position component, the planet can be randomly positioned in any quadrant, and in any position along its orbit.

This functionality was added to the code, and the resultant behaviour was identical to that seen in fig. 1 and fig. 2.

Position [x,y]	Velocity [v_x,v_y]
[1,1]	[-1,1]
[-1,1]	[-1,-1]
[-1,-1]	[1,-1]
[1,-1]	[1,1]

Table 1: Position parity vectors, with corresponding velocity parity vectors.

3.5 Moving Forward

As section 3.3 states, this code will be more than sufficient for initialising the first 8 time-steps of a predictor-corrector. Furthermore, with planets being positioned at completely random positions along their orbit, there is no chance of a deterministic effects arising from the initial setup of the system.

Hence, the project can continue towards the goal of creating a 4th Order Predictor Corrector. The logistics of this are detailed in the next section.

4 Project Plan

References

[1] Kokubo, E., and S. Ida
On Runaway growth of Planetesimals: N-body simulation
Icarus 1996, 123, 180–191.

[2] Barge, P., and R. Pellat
Mass spectrum and velocity dispersions during planetesimal accumulation. I. Accretion
Icarus 1991 93, 270?287

[3] Philip J. Armitage
Astrophysics of Planet Formation
Sec. 4.5.1, Pg. 131
10 Dec. 2009

[4] Williams, David
TPlanetary Fact Sheet - Metric
https://nssdc.gsfc.nasa.gov/planetary/factsheet/
As of: December 2, 2018

A Initial Conditions for Solar System

Planet	Mass (10^{24} kg)	Initial Orbital Velocity ($\mathrm{km\,s^{-1}}$)	Initial Orbital Radius (AU)
Earth	5.97	29.8	1.00
Mars	0.642	24.1	1.41
Jupiter	1898	13.1	5.03
Saturn	568	9.7	9.20
Uranus	86.8	6.8	18.64
Neptune	102	5.4	30.22

Table 2: Initial Conditions taken from [4]

B Relationship to previous projects

If you have already completed a project in the group where you are working, you should include a $\sim$ 1-page statement indicating how the PHY480 project differs from previous work. This is especially important if you are building on previous work done, for example, during a summer project. You must state clearly the new work that you have done during this semester.