Project Report - N-Body Simulation

Neil McBlane - s1305440

March 29, 2016

1 Tasks and Objectives

The three main objectives of this project were to:

- Write code to describe N-body systems interacting through Newtonian gravity.
- Introduce realistic conditions and simulate the Solar System.
- Compare simulation results to known astrophysical date.

2 Implementation

2.1 Final Code

2.1.1 Initial Processes

The program begins by taking the input particle file (XBody.txt) from the first command line argument and scanning through it to count the number of planets. It then creates a Particle3D array of the required length. Using a constructor implemented in the Particle3D class, another scanner then reads through the file to initialize a Particle3D for each address in the array corresponding to the planets of the input file. A third scanner then reads through the input parameter file (Parameters.txt) and stores each value: the gravitational constant, timestep length and the number of timesteps to run. Variables to store current time and an array to count the fraction of orbits complete for each planet were also initialized.

Three printWriters were then initialized for each of the files specified in the command line arguments: the VMD - a molecular visualization program [1] - trajectory file, the energy and total momentum check file and the orbit information file.

A number of initial properties of the system were then calculated. A Vector3D array was created and populated with the total force on each planet due to every other planet in the system - required as the Verlet time integration method takes an average of the "current" and "new" forces. A method defined in the Particle3D class was used to do so: it loops through each Particle3D and determines the Newtonian gravitational force due to every other Particle3D. To halve the required number of calculations, Newton's third law is invoked so that for every force vector $\vec{F_{ij}}$ added to the total force acting on Particle3D i, the corresponding equal and opposite vector $\vec{F_{ji}} = -\vec{F_{ij}}$ is added to the total force acting on Particle3D j.

The initial total momentum of the system was then determined, again using a method in the Particle3D class. This loops through each particle in the array and determines its momentum using $p_i = \frac{v_i}{m_i}$ then sums them to determine the total value. From this, another Particle3D method was used to adjust the total momentum of the system to zero by subtracting the Centre of Mass velocity of the system from the initial value of each planet.

To allow the accuracy of the simulation to be monitored, its total momentum and energy were calculated using Particle3D methods, with the former using that defined above. The latter summed the pairwise interaction between the planets and added this to the kinetic energy of each. As the energy between planet i and planet j is the same as the energy between j and i, the algorithm was structured such that if the interaction ij had already been considered then interaction ji was skipped. To achieve this, the algorithm effectively formed an upper triangular matrix, though it summed across each 'row' whilst running to reduce the number of stored values.

To determine the Aphelion and Perihelion (Apogee and Perigee for the Moon) of each planet, their initial positions were recorded in an array as a basis - with the Moon redefined so that its position was taken relative to

Earth. These were then used to populate a two-dimensional array which stored both the Aphelion and Perihelion and could be fed in to a Particle3D method later in the algorithm to check for changes.

The initial positions were printed to the trajectory file in the standard VMD file format using a method defined in Particle3D. The total energy and momentum of the system was also printed to the simulation checks file.

2.1.2 The Main Loop

The main loop, which is completed the number of times as specified by the parameters input file, begins here. In order to calculate the angular displacement of each planet during the course of one loop, the current position was recorded and stored in an array. The position and velocity of the planets were then updated using the Verlet time integration methods:

$$\vec{x}(t+dt) = \vec{x}(t) + \vec{v}(t)dt + \frac{1}{2m}\vec{F}(t)dt^2 \tag{1}$$

$$\vec{v}(t+dt) = \vec{v}(t) + \frac{1}{2m}(\vec{F}(t) + \vec{F}(t+dt))dt \tag{2}$$

Where $\vec{x}$ and $\vec{v}$ represent position and velocity respectively, dt is the size of the timestep (specified in the parameter input file), m is the mass of the planet and $\vec{F}$ is the force the planet experiences. To update an array of planet positions, a Particle3D method was called which calculated the forces between all the planets then applied this to Equation 1 for each planet in turn, a loop process implemented in another Particle3D method. The method which updates planet velocities again calculates the force due to these new positions ($\vec{F}(t+dt)$), and implements Equation 2 using another Particle3D method, which also takes in the forces prior to the position update ($\vec{F}(t)$). Following this update, the total energy and momentum of the system are then calculated.

A Particle3D method which compared the current planet positions (with the Moon taken relative to Earth) was then called. This compared the current magnitude of the planet's position to the recorded Aphelion and Perihelion (Apogee and Perigee for the Moon), stored in the two dimensional array initialised earlier. If the value was larger than the Aphelion or smaller than the Perihelion, it would replace what had been stored so far. Consideration of the dot product between the initial and updated position vectors of the loop allowed the fraction of an orbit completed by each planet during one loop to be determined. This was added to the current value recorded by the orbit counter for each planet. To do so, a method was implemented in Particle3D which performed the following calculation:

$$OrbitFraction = \frac{1}{2\pi}\frac{\vec{x}(t)\cdot\vec{x}(t+dt)}{|\vec{x}(t)||\vec{x}(t+dt)|} \tag{3}$$

The system time was then increased by the timestep value and - for every 100th timestep (i.e. once per day for our 0.01 day value) - the planet positions were printed to the trajectory file and the total energy and momentum printed to the checks file. The loop then repeated.

2.1.3 Final Processes

After the specified number of loops were completed, a number of characteristics of each orbit were calculated. A Particle3D method was implemented which calculated the eccentricity of each orbit using:

$$e = \frac{a-p}{a+p} \tag{4}$$

Where a is the orbital aphelion and p the perihelion. Kepler's third law was verified by comparing the LHS of equation 5 to the RHS:

$$\frac{T^2}{r^3} = \frac{4\pi^2}{GM} \tag{5}$$

Where T is the period of the orbit, r is the semi-major axis (easily determined from the orbital parameters), G is the gravitational constant and M is the mass of the body being orbited. A value of 1.000 indicated perfect agreement.

The recorded orbital parameters for each planet were then output to a file for easy comparison to the known values. All output files were closed and the program finished.

2.2 Changes From Design Document

A number of changes were made in the code from what was proposed in the design document. The most major being that every method defined outside of main in ParticleNBody was moved to Particle3D. This was done to reduce the bulk of ParticleNBody and to make Particle3D more flexible for potential future use.

Along with the methods moved over, additional ones were added. These accounted for the required functionality that was missing from the code outlined in the Design Document. Namely these were: centre of mass correction, orbit counter, aphelion/perihelion determination and Kepler III verification.

Changes to existing methods were also required. An orbit characteristics file was added to the command line arguments of ParticleNBody, along with the associated printwriter and print statements within the main method. This allowed key simulation values to be compactly presented for comparison with the real values. The main method was changed to include new functionality mentioned above, as well as calculations of initial velocity and forces to provide a base reference for aphelion/perihelion determination and for the Verlet step functions respectively. At the end of the main method, functionality was added to print the various orbital characteristics to file, as well as methods which closed the output files.

The methods moved to Particle3D were amended too. Those which calculated gravitational force and potential (and all which made use of these in turn) had a variable to take the value of the gravitational constant included as its value changed depending on the units chosen. The VMD printer was updated to take a Particle3D array as the design document only provided functionality for two. A printWriter was also included as an argument to make its appearance more elegant in the main method. The method to calculate the forces between a Particle3D array was updated to invoke Newton's third law, thereby reducing the required number of calculations. This was then placed in the array methods for Verlet Position and Velocity updates as each invoked the same loop in their calculations.

3 Results and Discussion

3.1 Three Body System

The first test of the code was that it reproduced the orbit of the simple two body system implemented in Exercise 3: this was found to work without issue. The next step was estimation of a simple system including the Sun, Mercury and Venus. Circular orbits were determined by appealing to conservation of energy and using the formula:

$$v = \sqrt{\frac{GM_{\odot}}{r}} \tag{6}$$

Where v is the speed of the planet, r was taken to be its mean orbital distance, G is the gravitational constant and $M_{\odot}$ is the mass of the Sun. All units used here were in SI. This produced the expected output, the period of which can be seen in Table 3.1. Taking values from NASA's Horizon system [3], a significant improvement in their accuracy was seen. As performance was satisfactory, we moved quickly on to the nine body system.

Body	Circular Period (days)	Elliptical Period (days)
Mercury	86.90	87.78
Venus	228.13	224.06

Table 1: A comparison of the orbital period obtained from circular estimation and true elliptical three body orbits

3.2 Nine Body System

Rearranging Equation 5 and applying it to Earth's allowed the value (and associated) of G to be determined:

$$G = \frac{4\pi^2 r^3}{MT^2} = 4\pi^2 AU^3 yr^{-2} {M_{\odot}}^{-1} = \frac{4\pi^2}{365.25^2} AU^3 day^{-2} {M_{\odot}}^{-1} \tag{7}$$

Therefore, by taking distances in units of AU, masses in units of $M_{\odot}$ and speeds in units of AU/day, we could operate with the sensible units for timestep as days. Again using NASA's Horizon system, the initial Solar System parameters were selected and (where required) converted to the aforementioned units. Their values were defined relative to the Solar System Barycentre as this was chosen as the origin.

3.2.1 Timestep Estimation

The Earth-Moon system was by far the most sensitive part of the simulation to changes in the timestep. As such, this was taken as the metric from which the maximum appropriate timestep was estimated. It was found anything larger than 0.01 days caused the Moon's orbit observed to deviate significantly from the expected value, and that decreasing the size further produced diminishing returns in terms of improvement to accuracy.

3.2.2 Orbit Characteristics

As can be seen in Figure 1, the simulation appears qualitatively accurate: we see the expected near circular orbits for all Planets in the same plane, as well as the highly eccentric and off-plane orbits of Halley's Comet (grey) and Pluto (brown).

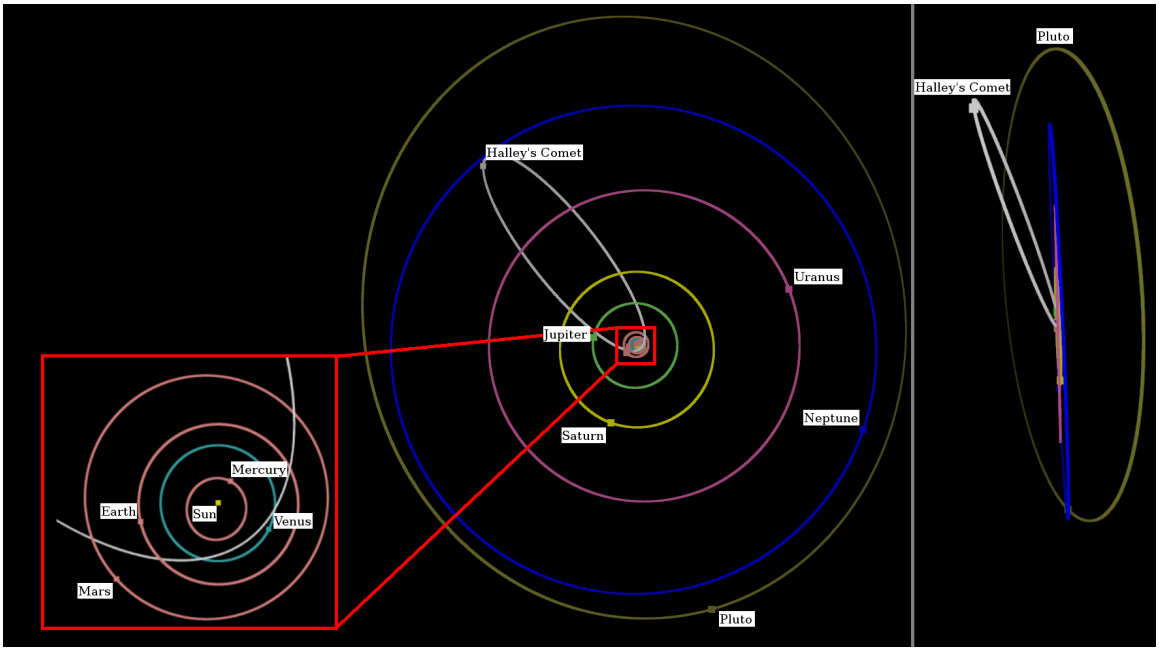

Figure 1: The trajectory of each planet in the 9 body system when run for 100,000 days.

Quantitatively, when compared to the observed Orbital characteristics of the Solar System (See Appendix, Table 4), we find our simulation to be very accurate (See Table 2). The number of orbits completed by majority of bodies is within 2% of the actual value. There is a slight trend for the error to increase as we travel further out in the system, but it is not to any great magnitude. This may be in some part due to the method we have used to count orbits: as we track only the fractional orbital displacement, different sections will be swept in varying times due to the slight eccentricities of the planetary orbits. The larger error in Halley's Comet (6.3%) is likely an exaggeration of this - due to its large eccentricity it will have a dramatic variation in speed. Perturbations due to its orbit's proximity to other planets may also have an effect, introducing error that our simulation isn't fully capable of reproducing successfully.

The aphelion and perihelion of each planet is highly accurate - each is comfortably within 2% of the true value. There appears to be little correlation between the accuracy of the former and latter, nor with orbital distance, which makes it difficult to determine what any simple source of this error could be. Eccentricity is a difficult metric to quantify the quality of our simulation by as its calculation greatly exaggerates the small errors we have on the aphelion and perihelion. This is most notable with Venus: despite our deviation for the aphelion and perihelion being only 1.1% and 1.4% respectively, it results in a 71% eccentricity change.

All four inner planets show a clear verification of Kepler's Third law, with a trend for the accuracy to decrease as planets further out are considered. In our formula we consider only the mass of the sun, but the other planets will play a role which increases in significance as we travel further out and therefore more mass is contained within each orbit - this is particularly prevalent for Pluto and Halley's Comet, both of which demonstrate a more significant deviation than most. The perturbations caused to the orbit of Halley's comet may also play a role

Body	Period (days)	Aphelion (AU)	Perihelion (AU)	Eccentricity	Kepler III
Mercury	87.78	0.471	0.303	0.218	1.000
Venus	224.06	0.736	0.708	0.019	1.000
Earth	364.30	1.021	0.976	0.022	1.000
Moon	27.26	0.003	0.002	0.067	0.987
Mars	685.40	1.665	1.378	0.094	1.000
Jupiter	4,329.00	5.453	4.945	0.049	0.999
Saturn	10,752.49	10.06	9.02	0.055	0.994
Uranus	30,303.03	20.11	18.14	0.052	0.998
Neptune	58,823.53	30.15	29.70	0.008	1.005
Pluto	90,909.09	48.58	29.41	0.246	1.023
HalleyC	25,641.03	35.13	0.576	0.968	1.181

Table 2: Orbit characteristics obtained by our simulation when run for 100,000 days (273.8 years).

in its large (18.1%) deviation, particularly if their effect is incorrectly considered due to the size of the timestep. The same can be said of the Moon: it showed particular sensitivity to timestep, only remaining bound below a value just above that which we selected. It may be in part due to it's proximity to Earth, and therefore requires accurate calculation to avoid being significantly affected by other bodies.

Perturbation of the orbit of Halley's Comet is also evidenced by a broadening of its trajectory in the outer reaches of the Solar System when the simulation is run for a significant time - around 1,000,000 days.

3.2.3 Energy and Momentum

Figure 2 shows the fluctuation in the total energy of the system, with a maximum spread of 10% from the initial value. This is a clear deviation, but it is not unreasonable and was found not to be sensitive to changes in timestep on the same order as the Moon orbit. Given the relative accuracy of the orbital characteristics of the system and the minute size of timestep required to minimise these fluctuations, it was felt that the timestep chosen was sufficiently small. The total momentum of the system remained at zero throughout the entire simulation, as was expected.

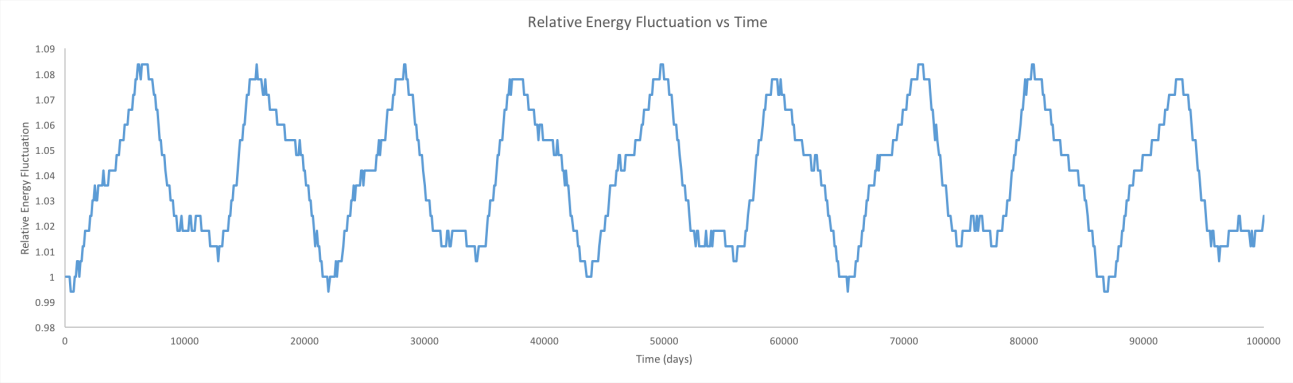

Figure 2: A graph demonstrating how the total energy of the system fluctuates relative to its initial value from day 1 to 100,000 of the orbit.

4 Software Testing and Comparisons

The implementation of the code received for testing differs from ours in a number of ways. Most notable is that many of their N-body methods, with the exception of the leapPosition and leapVelocity analogues, are included within the NBodyVerlet program - rather than being added on to the Particle3D class. This makes the code more bulky and reduces the flexibility of Particle3D for possible future use. They also do not have a toVMD method, instead then manually set up a loop to print a headed entry to file - this is again bulky. Instead of printing

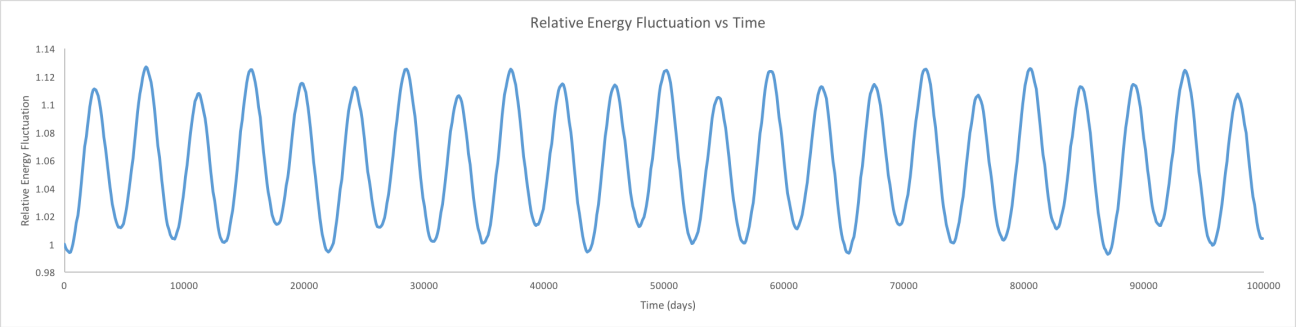

Figure 3: A graph demonstrating how the total energy of the assigned system fluctuates relative to its initial value from day 1 to 100,000 of the orbit.

their orbital characteristics to a file, they print them to terminal which makes later reference more difficult - they must be stored manually. Additionally, when calculating the total force they do not make use of the symmetry of Newton's third law and thus double the required calculations. They also print every loop which massively inflates the size of the output files.

There are a couple of features missing from this code: they do not determine aphelia and perihelia, eccentricities and do not verify Kepler's third law. Additionally, their orbit counter for the moon does not account for its orbit of Earth rather than the Sun. This therefore limits the number of metrics we have for comparing the code: leaving only orbital period, energy fluctuations and simulation time.

For 100,000 days (enough for a full Pluto orbit), their simulation takes two minutes. This is a significant improvement on ours, which takes five. The major influence behind this is that their timestep is significantly larger and therefore reduces the number of loops required by a factor of 12.5. A further reduction in calculations and therefore computational time is due to the fact that they do not calculate total momentum (which is perhaps not actually necessary) or check for aphelia/perihelia after every loop. A comparison of the simulation accuracy for the orbital period of each planet can be seen in Table 3. Despite a general trend of accuracy decreasing as orbit distance increases, their values are on the whole of slightly better accuracy than ours. This is particularly noteworthy given the much larger timestep value. The clear exception is the Moon: we cannot comment on its accuracy as its orbit is incorrectly defined. As Figure 3 demonstrates, their energy fluctuates about the same magnitude as ours.

Body	Our Period (days)	Their Period (days)	True Period (days)
Mercury	87.78	87.92	87.97
Venus	224.06	224.59	224.72
Earth	364.30	365.07	365.23
Moon	27.26	365.06	27.30
Mars	685.40	686.40	686.81
Jupiter	4,329.0	4,333.3	4,330.6
Saturn	10,753	10,779	10,759
Uranus	30,303	30,717	30,685
Neptune	58,824	60,036	60,190
Pluto	90,909	88,069	90,550
HalleyC	25,641	25,265	27,778

Table 3: Orbit periods obtained by ours and the assigned simulations when run for 100,000 days (273.8 years). True values taken from [3].

5 Conclusions and Post-Mortem

Overall myself and my partner (I hope) are pleased with the simulation and how we worked together to produce it. It is very satisfying to see a somewhat simple system produce such an accurate and visual output. I like the

layout of the code, and the elegance that moving our methods to Particle3D brought.

In hindsight, there are a couple of features I would like to have changed. As our orbits are all closed, the orbital period could have been calculated simply by looking at the time it takes to complete a simple orbit. This would have reduced the number of calculations required, but may not provide the most robust indicator of the total number of orbits completed. Additionally, the method we used does not provide an accurate measurement of fractional orbits for highly eccentric bodies. This is due to their variable speed, and could have been accounted for by considering Kepler's Second Law. As the energy and momentum checks are calculation-heavy and not always required, perhaps making these optional via a command line interface would have made running the code more convenient. It would have also been preferable to implement checks for the Moon in a more elegant manner, at the moment a number of exceptions have to made within loops.

In terms of group work, the changes I would make are purely administrative. We spread the workload well and were able to work on different areas of the code concurrently using collabedit.com [4]. We did not use versioning well, maintaining only the latest copy on our personal computers and on a drive folder. Luckily we did not run in to any issues pertaining to this, but using a service such as GitHub [5] would have been an improvement.

All in all, I am very happy with how this project has gone.

appendix

True Astronomical Data

Body	Orbit Period (days)	Aphelion (AU)	Perihelion (AU)	Eccentricity	Kepler III
Mercury	87.968	0.467	0.307	0.206	1.000
Venus	224.695	0.728	0.718	0.017	1.000
Earth	365.242	1.017	0.983	0.017	1.000
Moon	27.32	0.003	0.002	0.055	0.987
Mars	686.973	1.666	1.381	0.093	1.000
Jupiter	4,330.595	5.455	4.950	0.048	0.999
Saturn	10,746.94	10.09	9.02	0.056	0.994
Uranus	30,588.74	20.11	18.33	0.046	0.998
Neptune	59,799.9	30.33	29.81	0.009	1.005
Pluto	90,560	49.32	29.66	0.249	1.023
HalleyC	27,356.75	35.1	0.586	0.967	1.181

Table 4: NASA observed astronomical data [2]

References

[1] Ks.uiuc.edu. (2016). *VMD - Visual Molecular Dynamics.* [online] Available at: `http://www.ks.uiuc.edu/Research/vmd/` [Accessed 14 Mar. 2016].

[2] Pds.jpl.nasa.gov. (2016). *Welcome to the Planets.* [online] Available at: `https://pds.jpl.nasa.gov/planets/` [Accessed 21 Mar. 2016].

[3] Ssd.jpl.nasa.gov. (2016). *HORIZONS Web-Interface.* [online] Available at: `http://ssd.jpl.nasa.gov/horizons.cgi` [Accessed 19 Feb. 2016].

[4] Collabedit.com. (2016). *online text editor - collabedit.* [online] Available at: `http://collabedit.com/` [Accessed 13 Jan. 2016].

[5] GitHub. (2016). *Build software better, together.* [online] Available at: `https://github.com/` [Accessed 9 Mar. 2016].